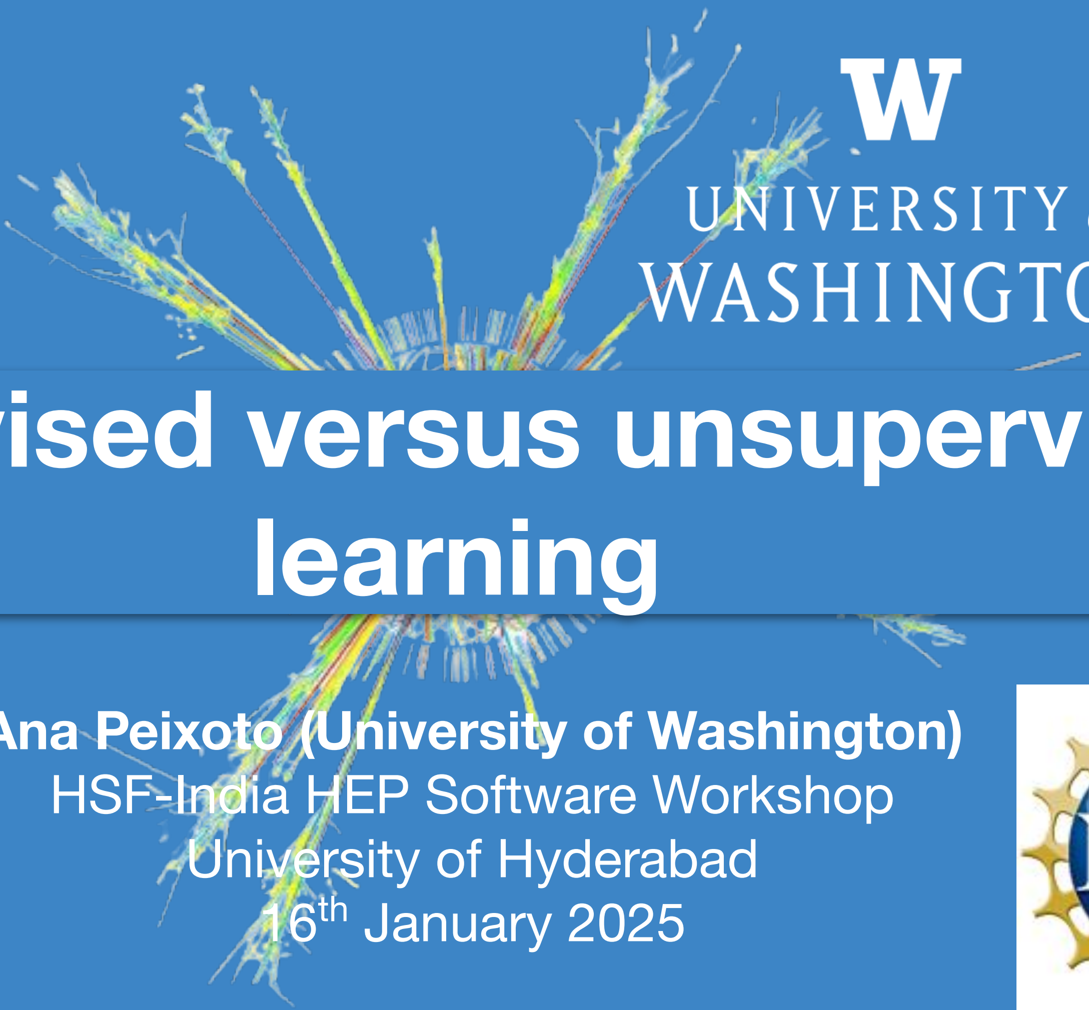


Supervised versus unsupervised learning

Ana Peixoto (University of Washington)

HSF-India HEP Software Workshop

University of Hyderabad

16th January 2025

Reminder about Machine Learning

Huge variety of choice in goals, formulations, training procedures, …

- Mathematical structure of model: (deep) neural networks, convolutional networks, transformer models, (boosted) decision trees, and many others
- Input data: supervised (learn from simulation with truth-labels) and unsupervised learning (learn from data without truth labels)
- Learning goals: classification, regression
- Scope of model: discrimination or generative

Classes of Statistical Learning algorithms

Supervised:

- If we know the probability density of S and B, or if at least we can estimate it
- E.g. we use "labeled" training events ("Signal" or "Background") to estimate p(x|S), p(x|B) or their ratio

Semi-supervised:

- It has been shown that even knowing the label for part of the data is sufficient to construct a classifier

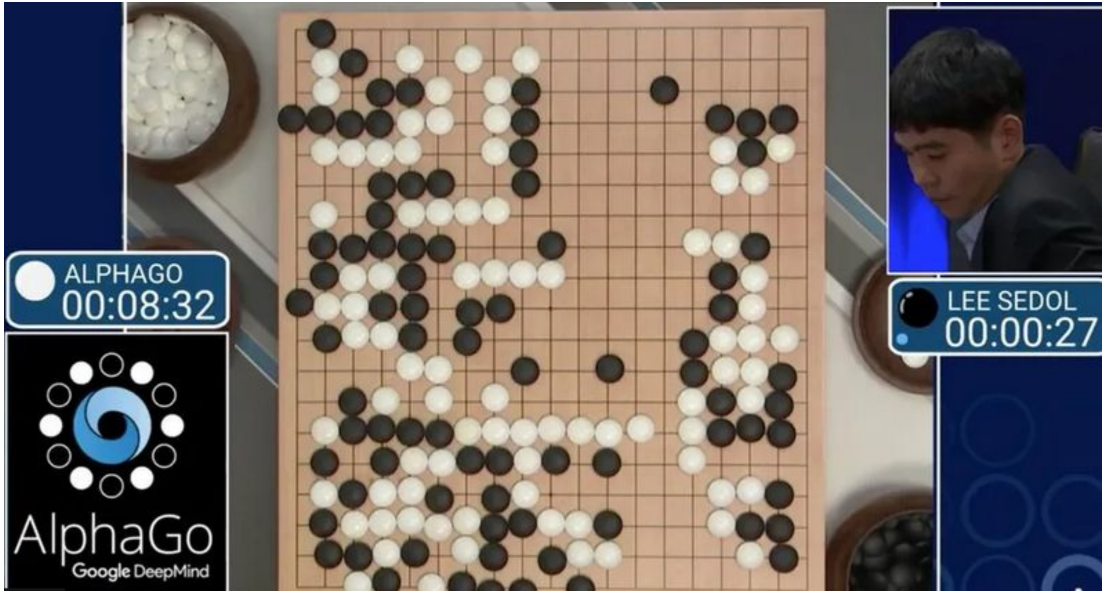
Unsupervised:

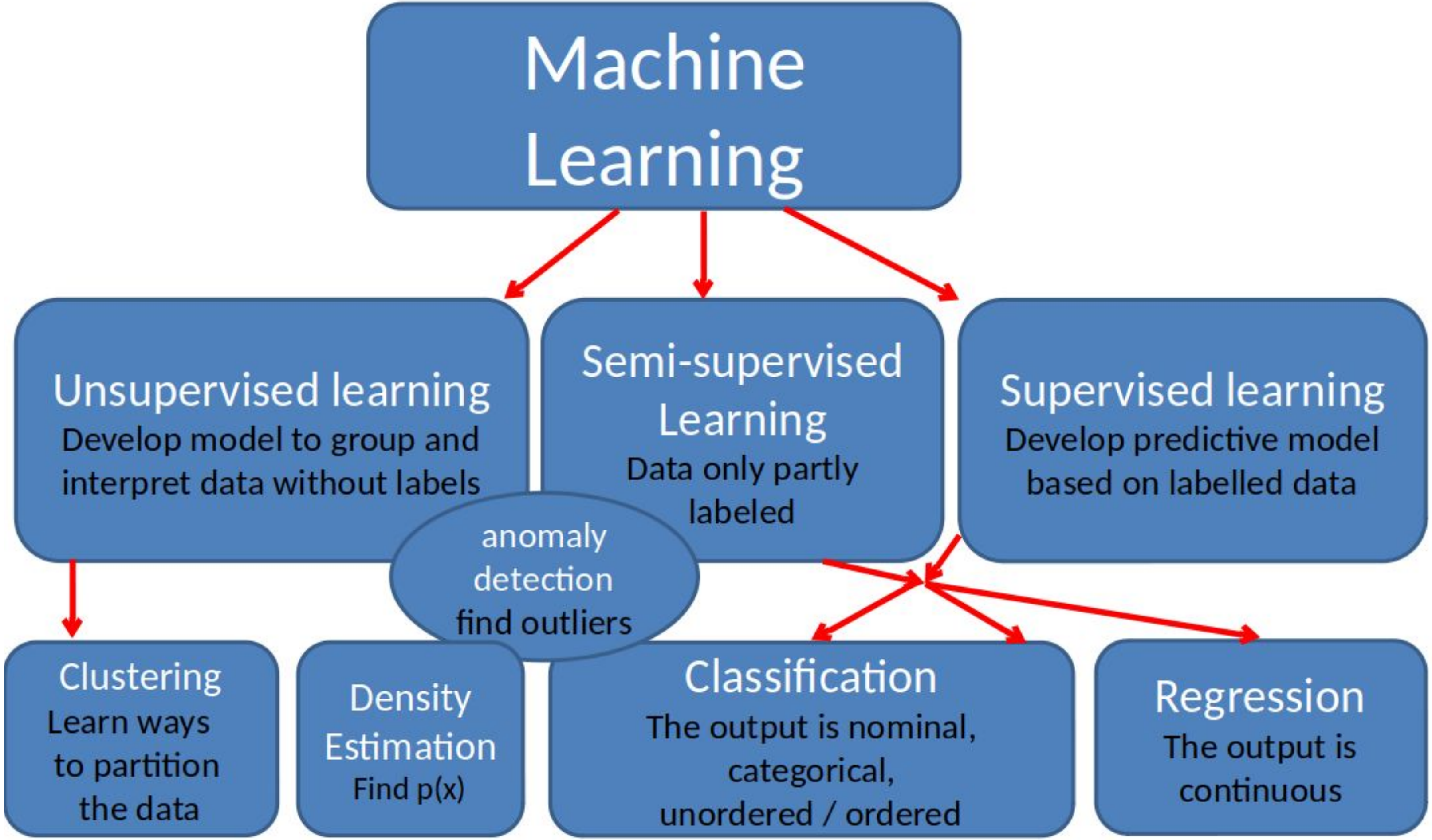
- If we lack an a-priori notion of the structure of the data, and we let an algorithm discover it without e.g. labeling classes cluster analysis, anomaly detection, unconditional density estimation.

Classes of Statistical Learning algorithms

We may also single out **reinforcement learning**

- The algorithm learns from the success or failure of its own actions
- E.g. a robot reaches its goal or fails





Machine Learning

Unsupervised learning
Develop model to group and interpret data without labels

Semi-supervised Learning
Data only partly labeled

Supervised learning
Develop predictive model based on labelled data

anomaly detection
find outliers

Clustering
Learn ways to partition the data

Density Estimation
Find p(x)

Classification
The output is nominal, categorical, unordered / ordered

Regression
The output is continuous

Supervised Learning

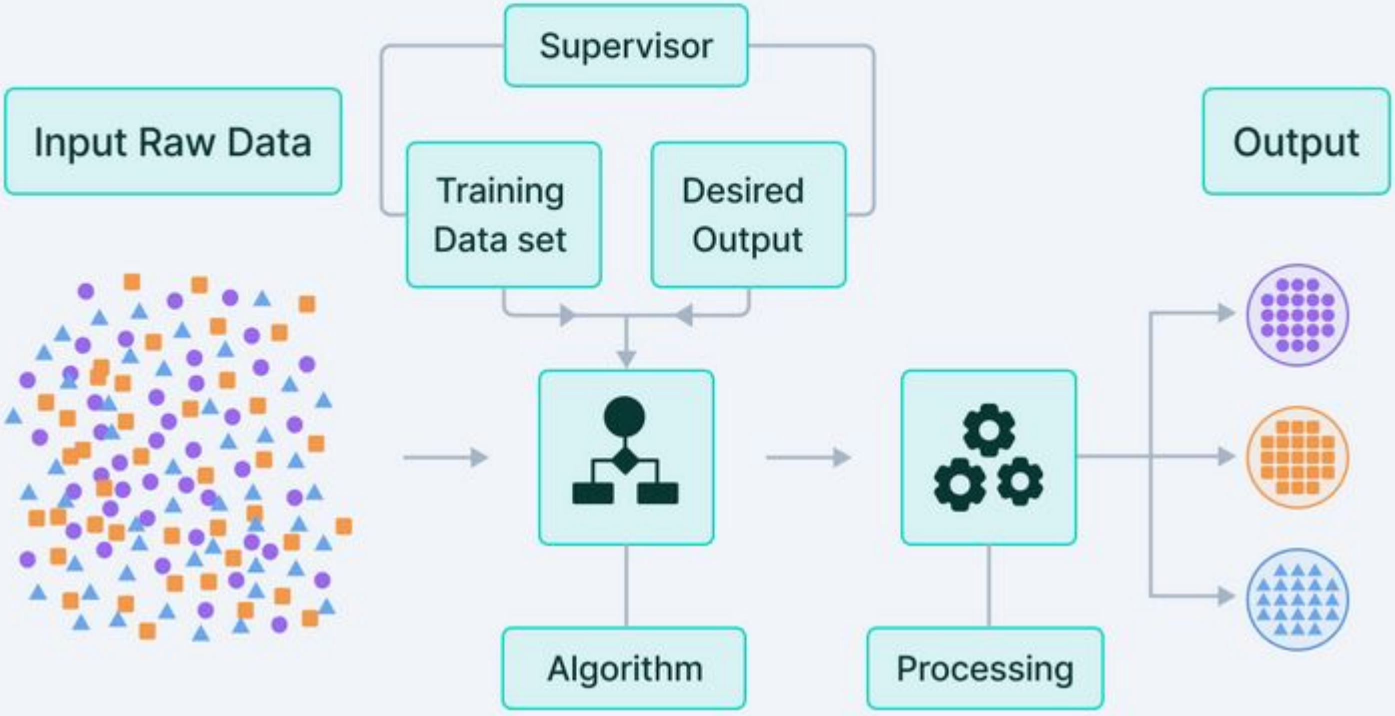
Starting point:

- A vector of n predictor measurements X (inputs, regressors, covariates, features, independent variables)

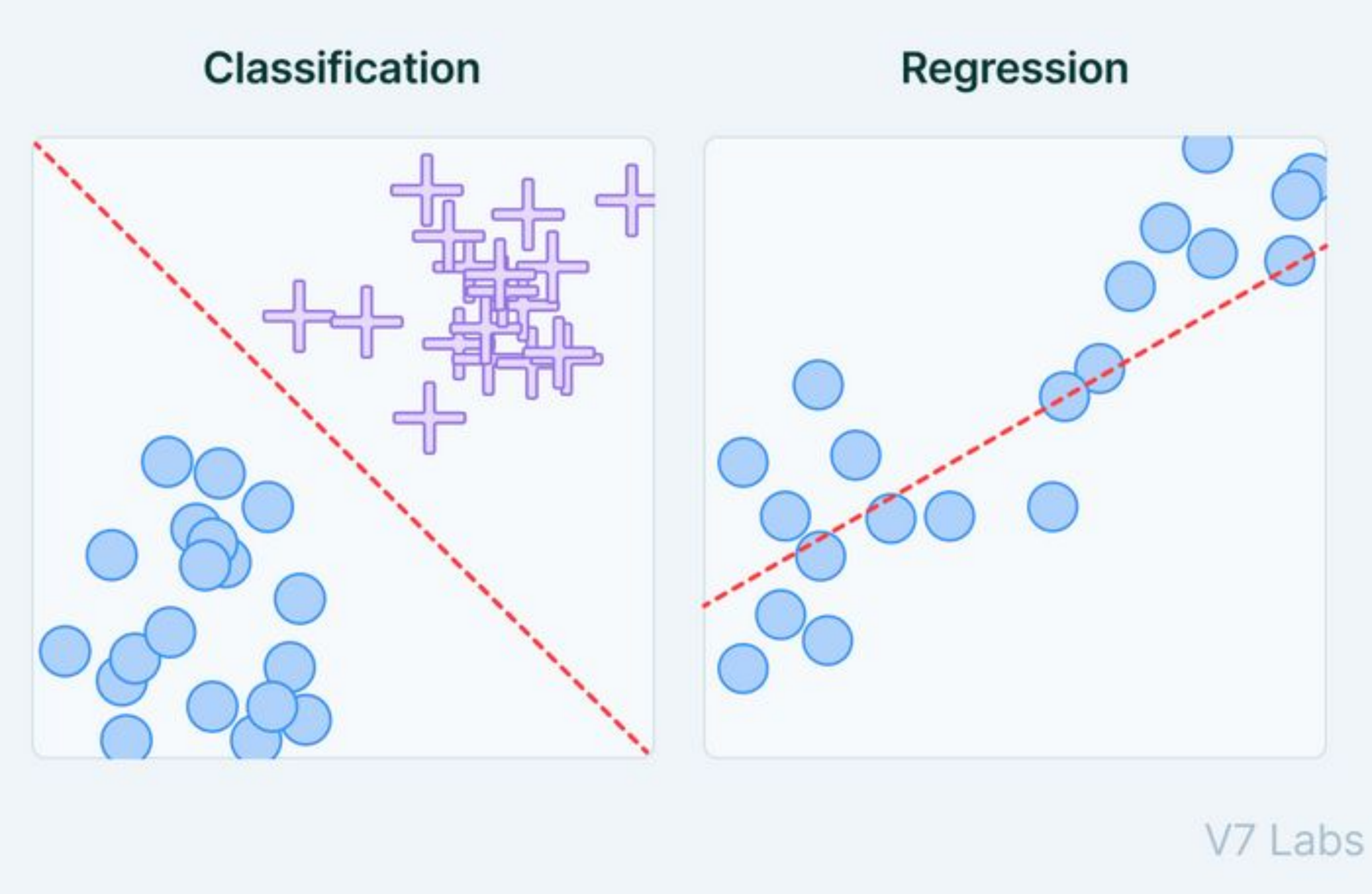
- One has training data {(x,y)}: events (or examples, instances, observations...)

- The outcome measurement Y (dependent variable, response or target)
  - In classification problems Y can take a discrete, unordered set of values (signal/background, index, type of class)
  - In regression, Y has a continuous value

Supervised Learning



V7 Labs


Classification
Regression
V7 Labs

Supervised Learning

Objective:

- Using the data at hand, we want to predict y* given x*, when (x*,y*) does not necessarily belong to the training set.
- The prediction should be accurate: |f(x*)-y*| must be small according to some useful metric (see later)

We would like to also:

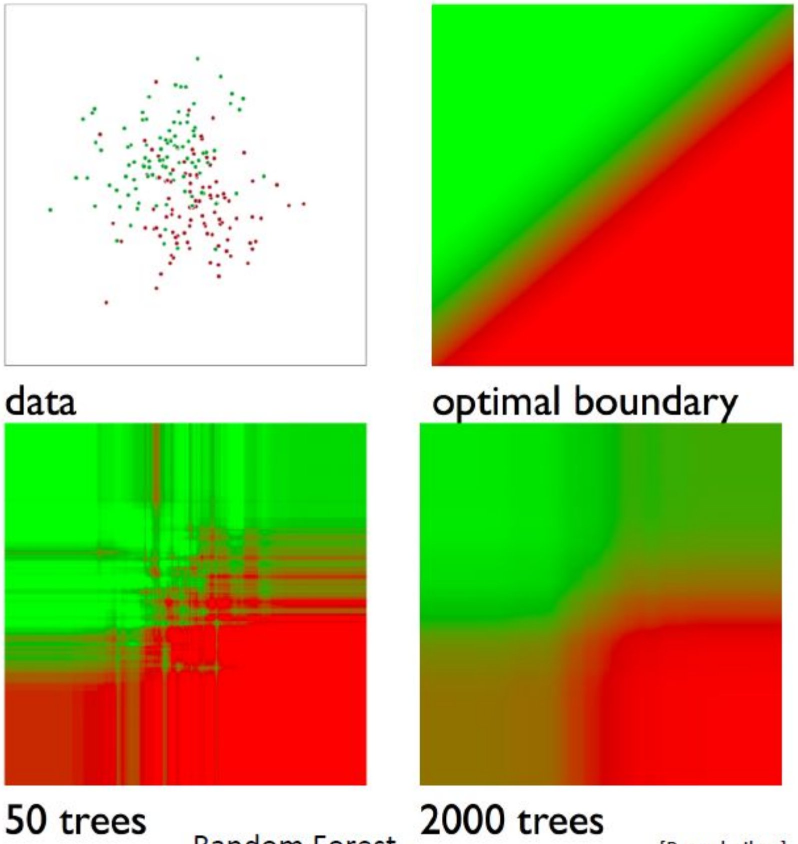
- Understand what feature of X affects the outcome
- How assess the quality of our prediction

Random forests

- Random Forest algorithm was first proposed by Breiman (2001), but is based on a 1995 idea of Tim Kan Ho
- RF employs two ensemble techniques: The first is bagging of the training sample, to grow a forest of different trees based on different training data. The second is the subsampling of the feature space.
- If I choose a subset of the variables (e.g. x1, x3, x7) to create a split in a node of a decision tree, and another subset (x2, x4, x5, x7) to create a different one, there will be events that get classified in a different way by the two nodes
- Often there is a dominant variables that is used to decide the split, offsetting the power of the subdominant ones. RF avoids that problem by reducing the correlation of different trees

Random forests

- Tree ensembles (like the Random Forest algorithm) have a number of attractive properties
  - usually do not overfit
  - powerful learners
- In addition they retain the advantages of DTs:
  - simple to understand and interpret
  - easy to train
  - work equally well with continuous as well as categorical data types
  - no need to pre-process the data (e.g. invariant to standardization)



data

optimal boundary

50 trees

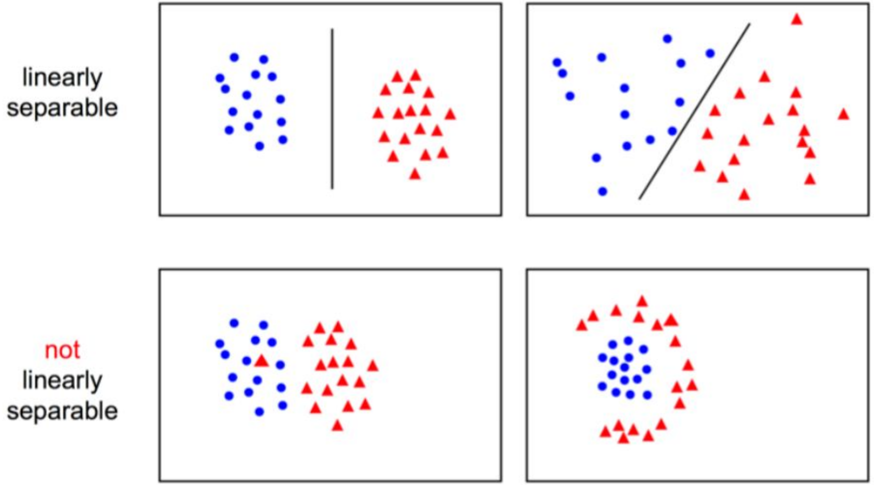
2000 trees

Random Forest

[Rogozhnikov]

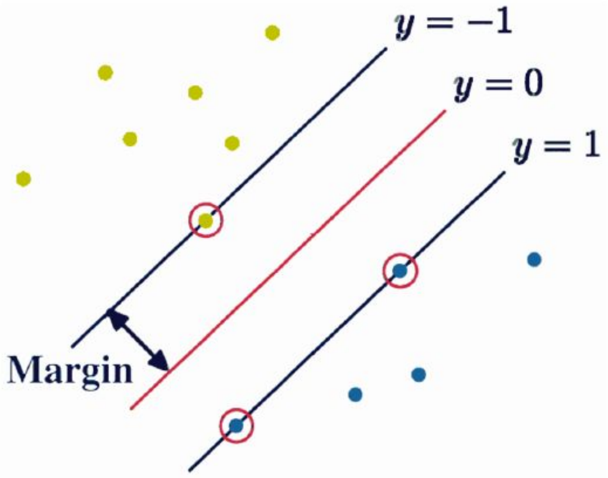
Support vector machines

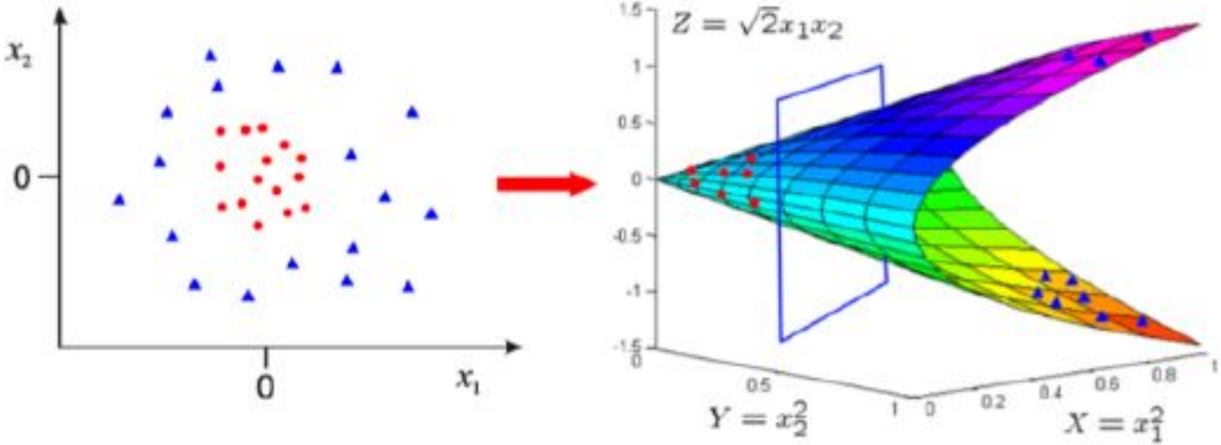
- Binary linear classifier that tries to find the best separation between two classes of data
- Basic concept can be again explained with linear separation between the classes (a hyperplane in the feature space)
- Non-linear separation is possible by extending the technique to additional dimensions (more complex data)



Support vector machines

- Which hyperplane to use?
- A linear SVM is the simplest hypothesis but a non-linear SVM can provide better results

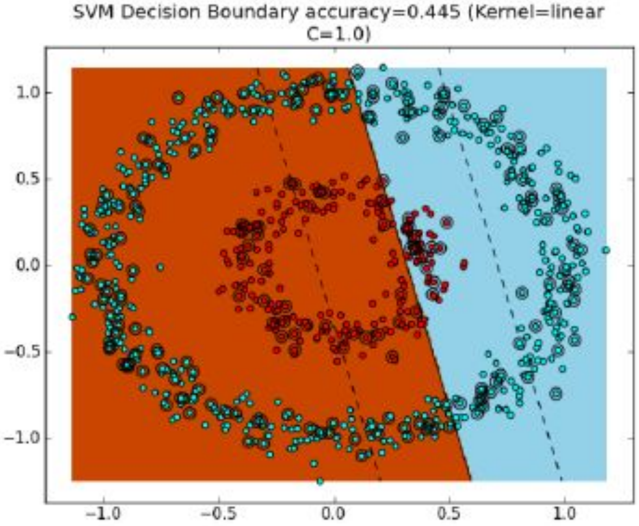




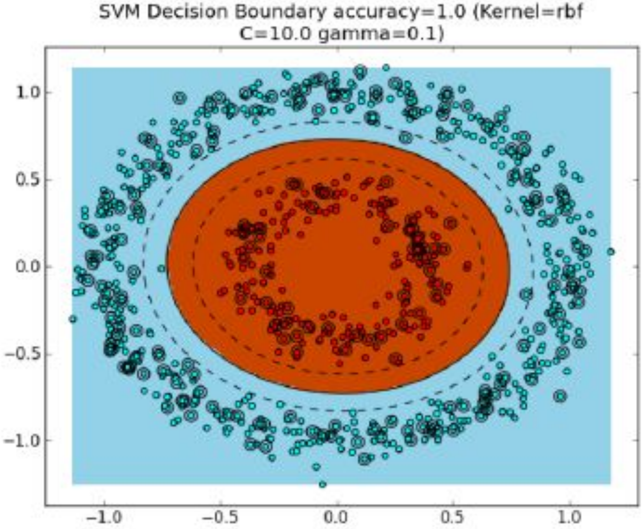
Support vector machines

- Which hyperplane to use?
- A linear SVM is the simplest hypothesis but a non-linear SVM can provide better results
- Kernel trick: by transforming the data with a map ϕ(x), we may find a hyperplane in the larger dimensions space where the classes are linearly separable

Linear kernel



SVM Decision Boundary accuracy=0.445 (Kernel=linear C=1.0)

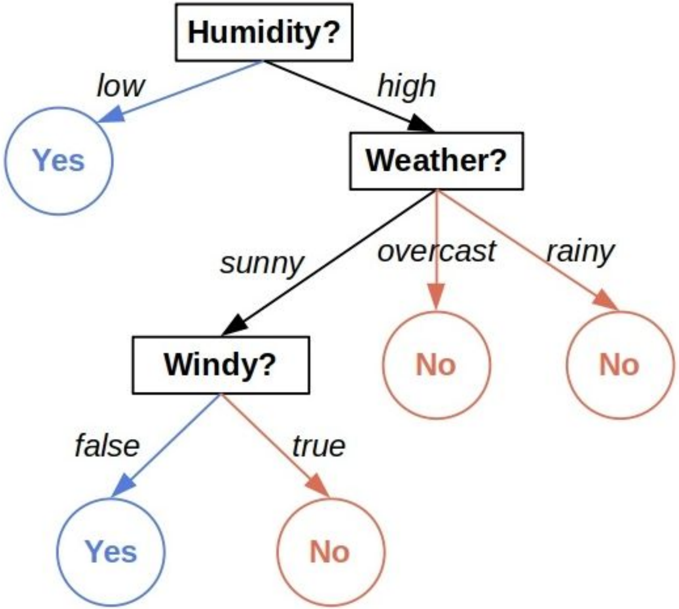


SVM Decision Boundary accuracy=1.0 (Kernel=rbf C=10.0 gamma=0.1)

Gaussian kernel

Boosted Decision Trees

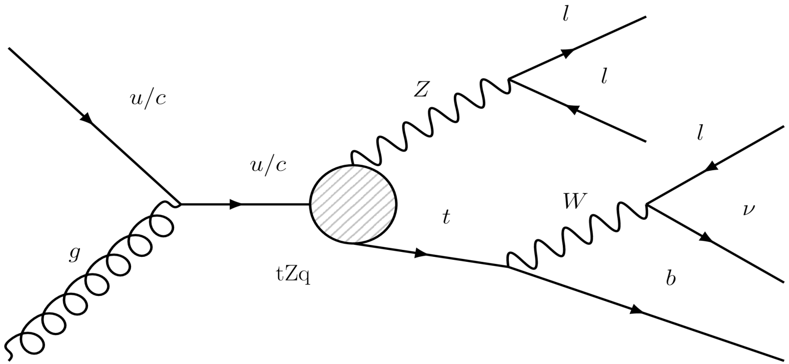
- A "decision tree" is a tree constructed by "leaves" that are rules to split the data in the different classes, based on the data features
- If each event to be classified has variables x_1, x_2, x_3, x_4,... (I can create a tree by posing conditions on each variable, in a chain)
- Idea of Boosting is instead to train a sequence of models, each of which gives more weight to events not classified correctly by the previous ones



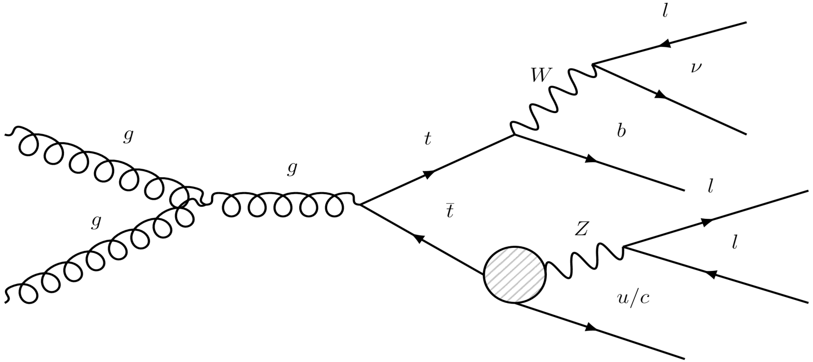
Decision tree explanation

Boosted Decision Trees

- Top quark decay via FCNC processes present a powerful probe of new physics and it can occur in two modes:
  - In **production**: ***t+X*** production with X = *H*, *Z*, *g*, *γ*
  - In **decay**: ***tt*** production (with ***t*** →***qX***) with *q* = *u*, *c*

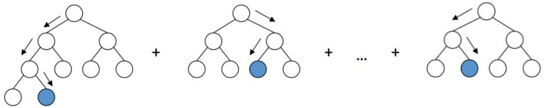


Feynman diagram at Leading-Order for FCNC *tZ* production

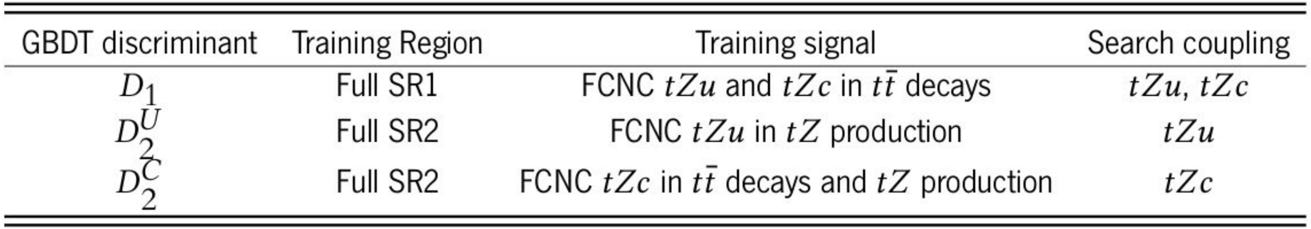


Feynman diagram at Leading-Order for FCNC *tt* decay

Boosted Decision Trees



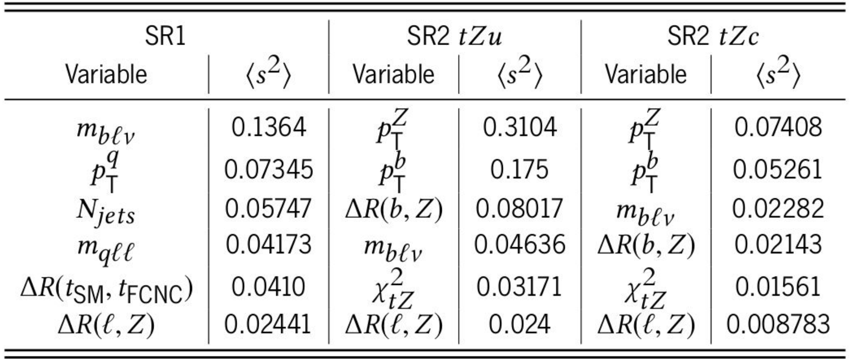
- **Three multivariate discriminants** defined using **Gradient Boosted Decision Trees** (GBDT): FCNC *tZu* and *tZc* in decay, FCNC *tZu* in production and FCNC *tZc* in decay and production



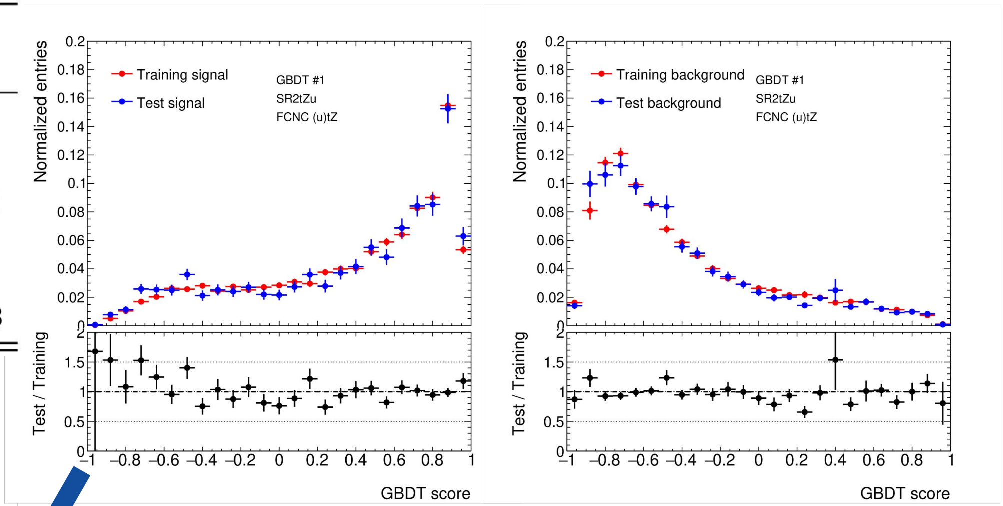
GBDT discriminant	Training Region	Training signal	Search coupling
D_1	Full SR1	FCNC tZu and tZc in $t\bar{t}$ decays	tZu, tZc
D_2^U	Full SR2	FCNC tZu in tZ production	tZu
D_2^C	Full SR2	FCNC tZc in $t\bar{t}$ decays and tZ production	tZc

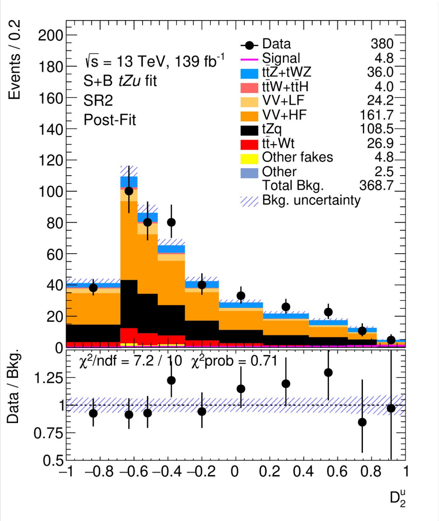
- All the signal and background events selected by the signal regions divided into five equal parts and used for the training (80%) or for the testing (20%)
- Choice of the input variables taking into account the separation power, the correlation with other variables and the performance loss

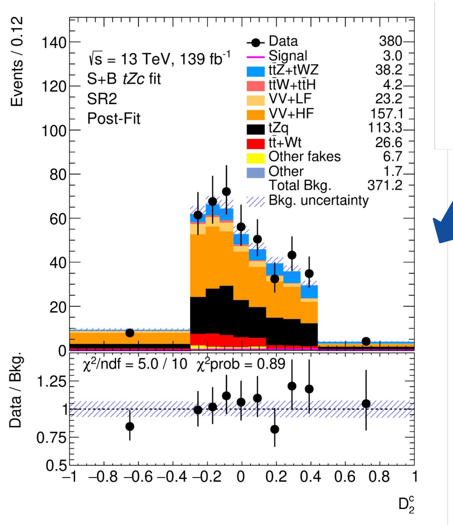
Boosted Decision Trees



SR1		SR2 tZu		SR2 tZc	
Variable	$\langle s^2 \rangle$	Variable	$\langle s^2 \rangle$	Variable	$\langle s^2 \rangle$
$m_{b\ell\nu}$	0.1364	p_T^Z	0.3104	p_T^Z	0.07408
p_T^q	0.07345	p_T^b	0.175	p_T^b	0.05261
N_{jets}	0.05747	$\Delta R(b, Z)$	0.08017	$m_{b\ell\nu}$	0.02282
$m_{q\ell\ell}$	0.04173	$m_{b\ell\nu}$	0.04636	$\Delta R(b, Z)$	0.02143
$\Delta R(t_{SM}, t_{FCNC})$	0.0410	χ^2_{tZ}	0.03171	χ^2_{tZ}	0.01561
$\Delta R(\ell, Z)$	0.02441	$\Delta R(\ell, Z)$	0.024	$\Delta R(\ell, Z)$	0.008783





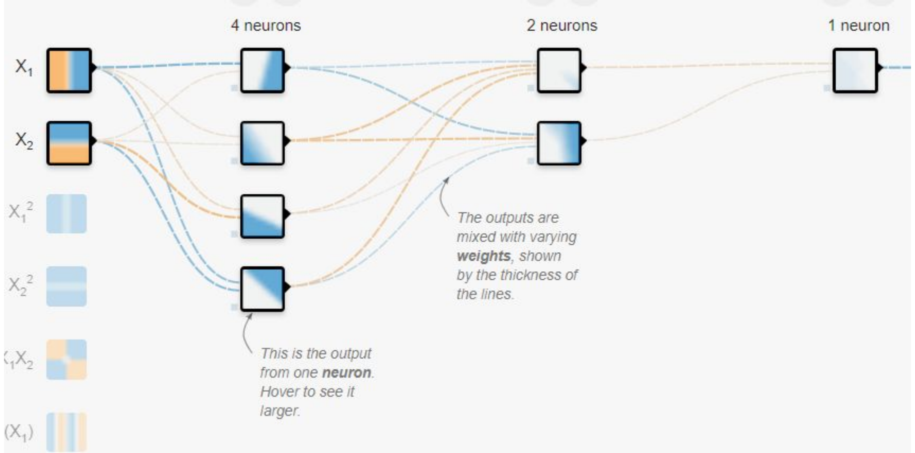


Training and test comparison for signal and background for the discriminant with *tZu* coupling on production

PRD 108, 032019 (2023)

Neural Networks

- A neural network (NN) is a program that simulates the behaviour of a series of neurons and their connections
- NNs are capable of producing very flexible functions of the feature space variables
- At the heart of the NN there is an architecture of nodes organized in layers. Every "neuron" of a layer receives inputs from some of (or all) the neurons of the previous layer

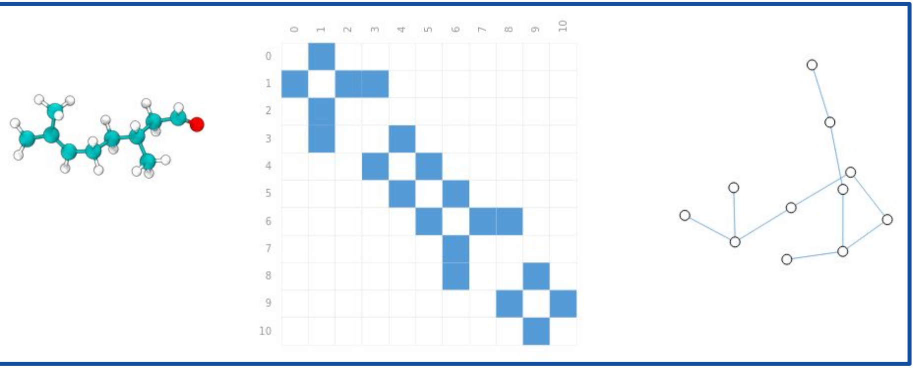


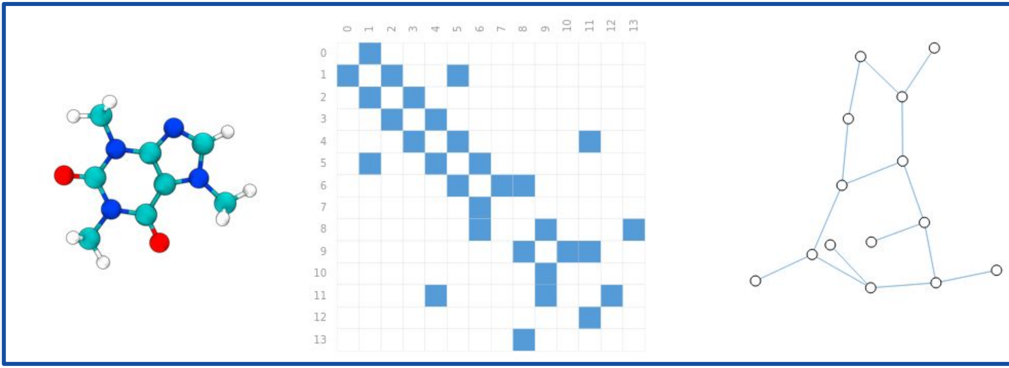
*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

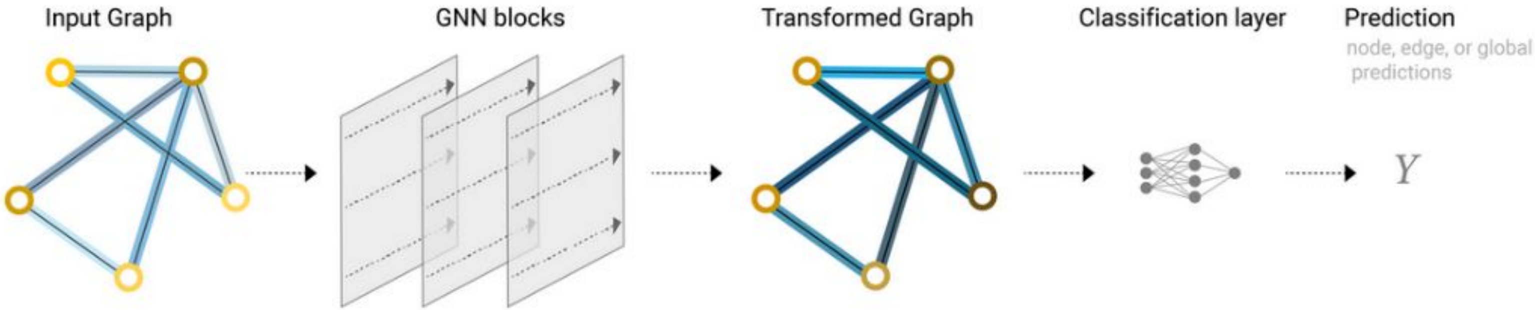
*This is the output from one **neuron**. Hover to see it larger.*

Graph Neural Networks

- NNs that can be represented as graphs







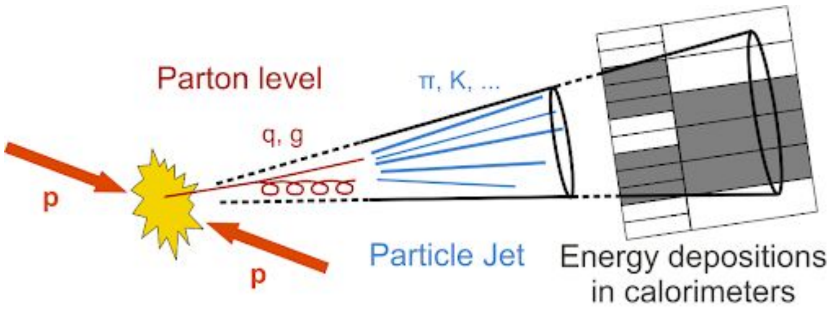
Suggested reading: https://distill.pub/2021/gnn-intro/

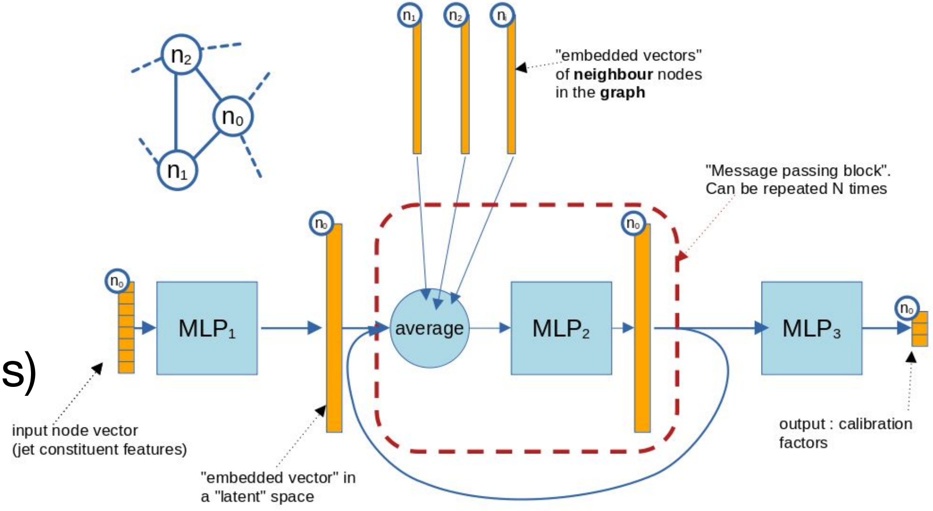
Graph Neural Networks

- But jets can also be represented as a graph!

- Calibration of jet constituents → Better reconstruction of the energy flow details

- Using Graph Neural Networks to calibrate jet constituents: performing node-level (constituents) regression from graph-level (jets) constraints



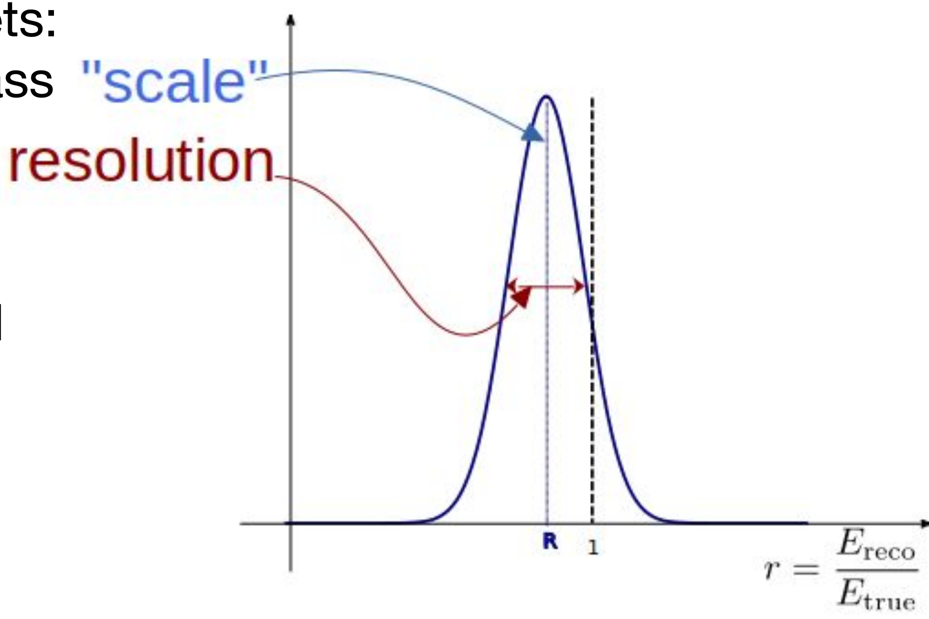

n_2
n_0
n_1
n_1
n_2
n_i
"embedded vectors" of **neighbour** nodes in the **graph**
"Message passing block". Can be repeated N times
n_0
MLP$_1$
n_0
average
MLP$_2$
n_0
MLP$_3$
n_0
input node vector (jet constituent features)
"embedded vector" in a "latent" space
output : calibration factors

Graph Neural Networks

Evaluation of the calibration performances with R=1.0 and R=0.4 jets:

- Check physics jets energy and mass response
- Rebuild jets with GNN calibrated constituents
- Distributions of ratio E_{calib}/E_{true} and M_{calib}/M_{true}
- Consider scale and resolution

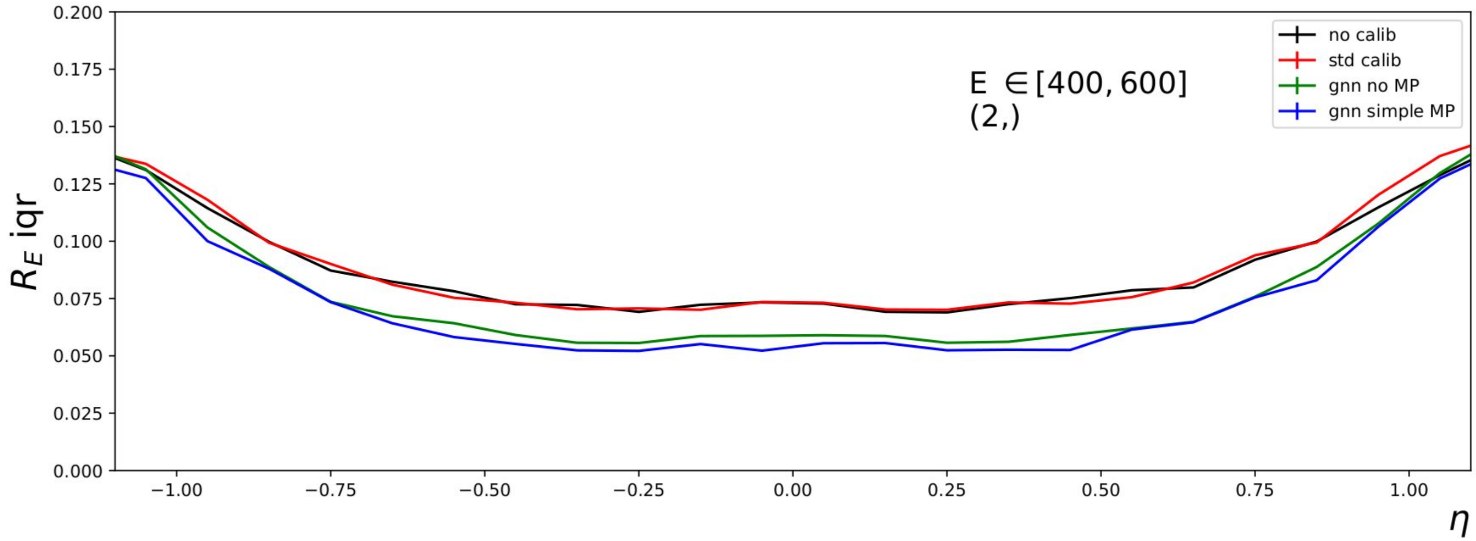
⇒ Get this response in many energy and/or mass bins!



Graph Neural Networks

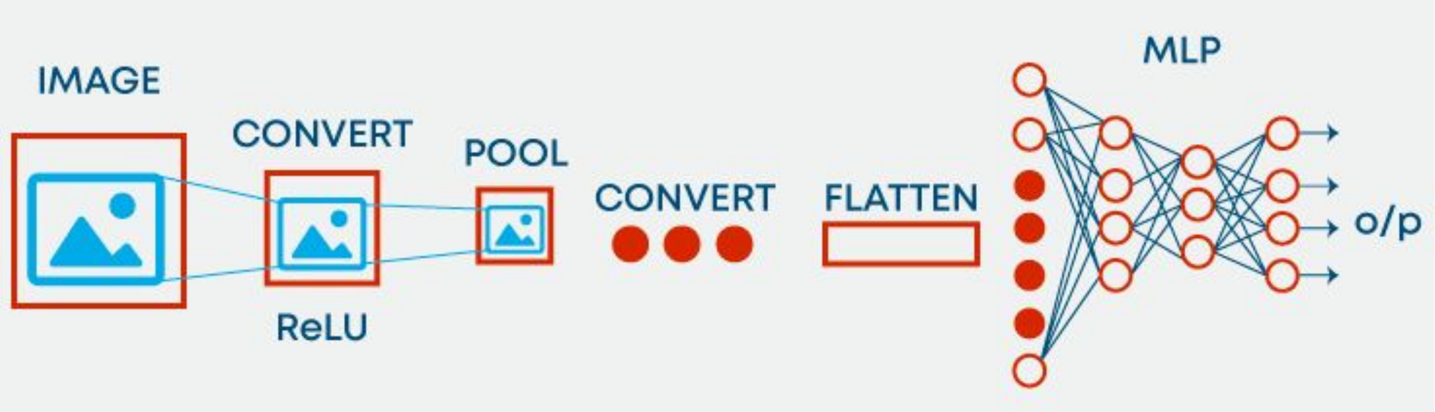
For the case of ATLAS:

- Energy scale is well reconstructed - almost as well as standard ATLAS calibration
- Energy resolution is much improved!



Convolutional Neural Networks

- Contains a three-dimensional arrangement of neurons instead of the standard two-dimensional array
- Each neuron in the convolutional layer (1st) only processes the information from a small part of the visual field
- Understands the images in parts and can compute these operations multiple times to complete the full image processing



Convolutional Neural Networks

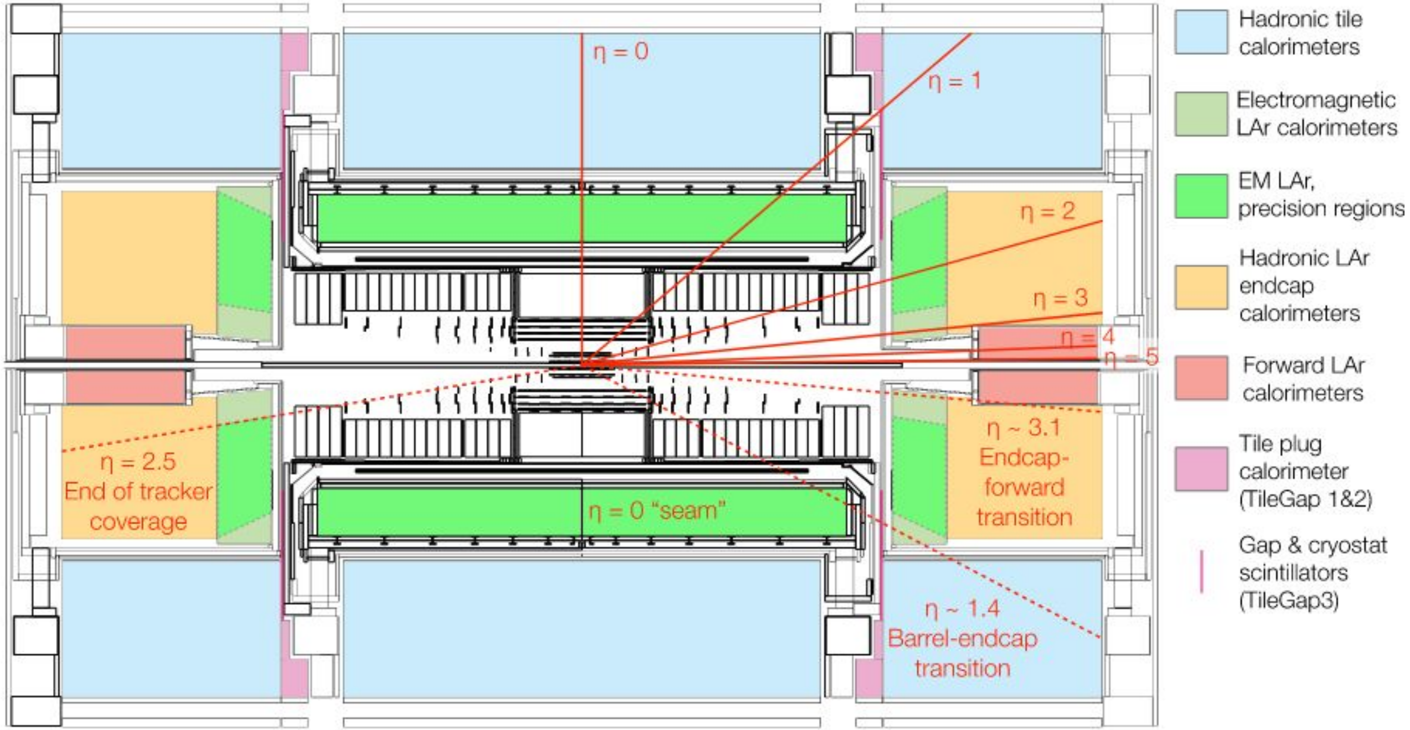
Applications:
Image processing

- Computer Vision (the usual cats vs dogs challenge)
- Speech Recognition
- Machine translation

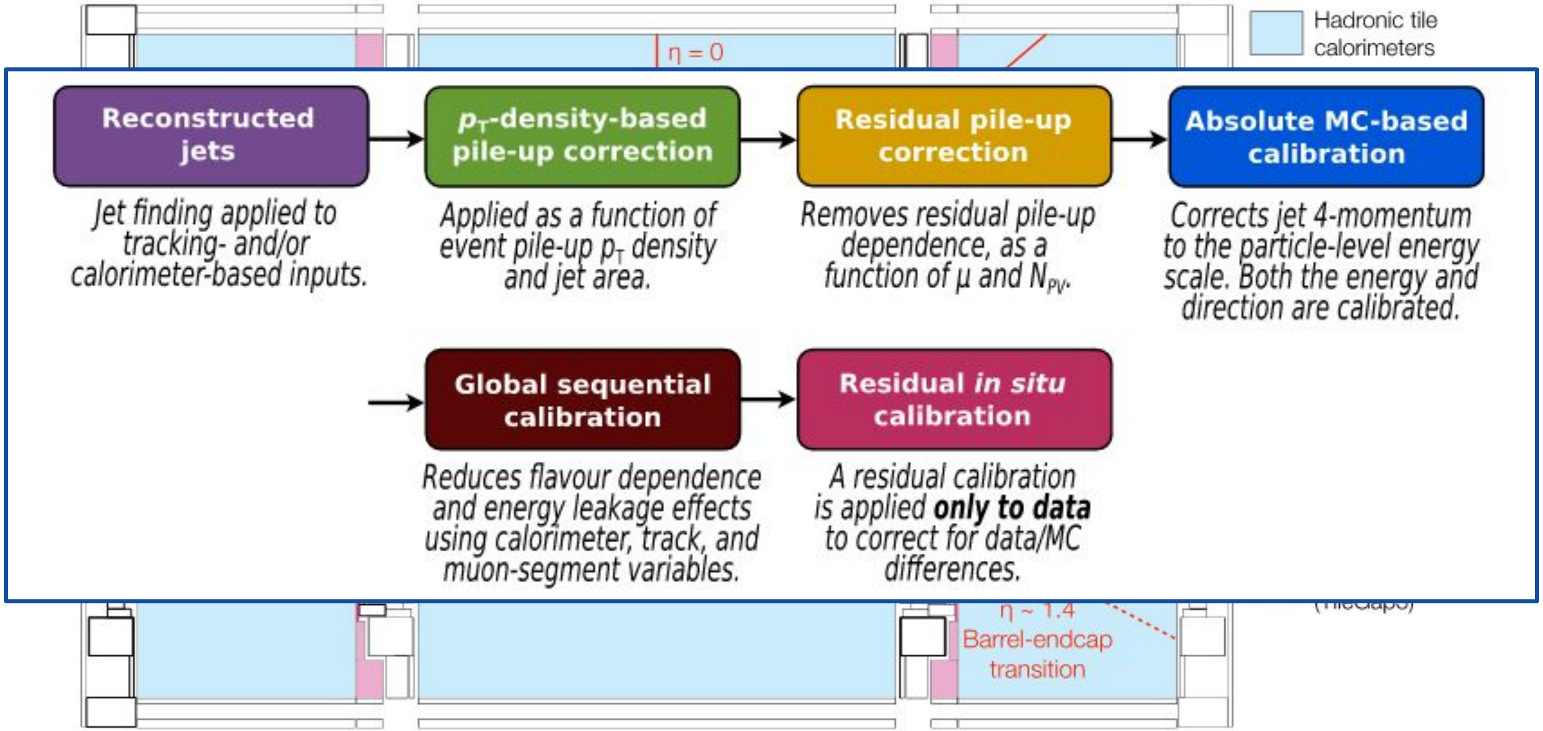
In HEP:

- Calibration of jets again…
- And much more!

Convolutional Neural Networks


η = 0
η = 1
η = 2
η = 3
η = 4
η = 5
η = 2.5
End of tracker coverage
η = 0 “seam”
η ~ 3.1
Endcap-forward transition
η ~ 1.4
Barrel-endcap transition
Hadronic tile calorimeters
Electromagnetic LAr calorimeters
EM LAr, precision regions
Hadronic LAr endcap calorimeters
Forward LAr calorimeters
Tile plug calorimeter (TileGap 1&2)
Gap & cryostat scintillators (TileGap3)

Convolutional Neural Networks



Reconstructed jets → **p_T-density-based pile-up correction** → **Residual pile-up correction** → **Absolute MC-based calibration** → **Global sequential calibration** → **Residual *in situ* calibration**

Reconstructed jets: *Jet finding applied to tracking- and/or calorimeter-based inputs.*

p_T-density-based pile-up correction: *Applied as a function of event pile-up p_T density and jet area.*

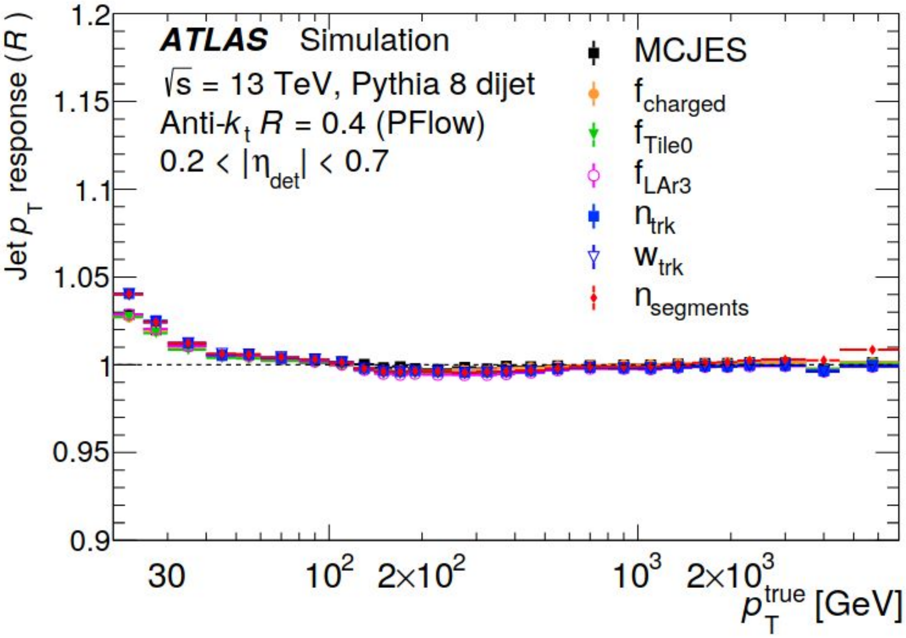
Residual pile-up correction: *Removes residual pile-up dependence, as a function of μ and N_{PV}.*

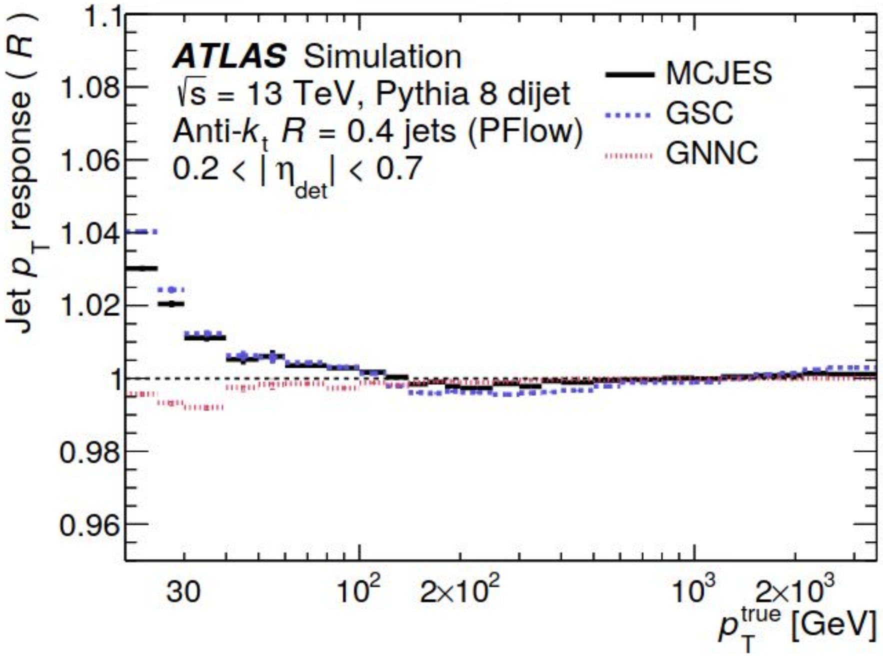
Absolute MC-based calibration: *Corrects jet 4-momentum to the particle-level energy scale. Both the energy and direction are calibrated.*

Global sequential calibration: *Reduces flavour dependence and energy leakage effects using calorimeter, track, and muon-segment variables.*

Residual *in situ* calibration: *A residual calibration is applied **only to data** to correct for data/MC differences.*

Convolutional Neural Networks



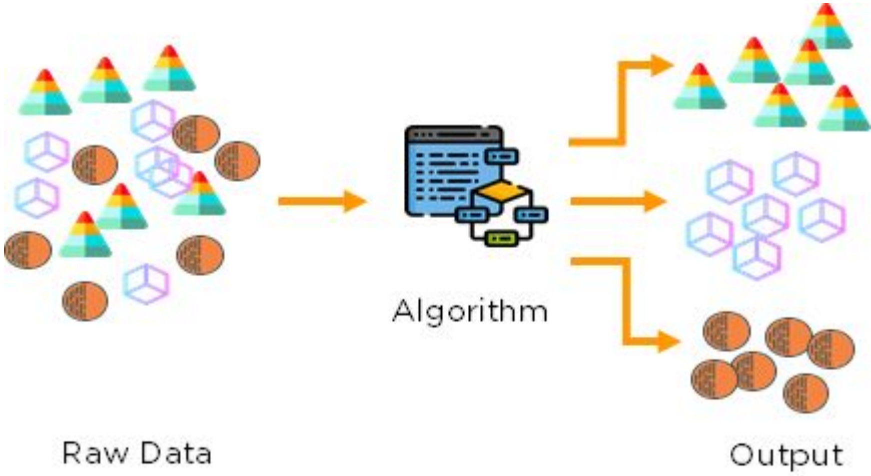


“New techniques for jet calibration with the ATLAS detector”

Unsupervised Learning

Starting point:

- A vector of n predictor measurements X (inputs, regressors, covariates, features, independent variables).
- One has training data {x}: events (or examples, instances, observations...)
- There is no outcome variable Y



Unsupervised Learning

Objective is much fuzzier:

- Using the data at hand
- Find groups of events that behave similarly
- Find features that behave similarly
- Find linear combinations of features exhibiting largest variation

Challenges:

- Hard to find a metric to see how well you are doing

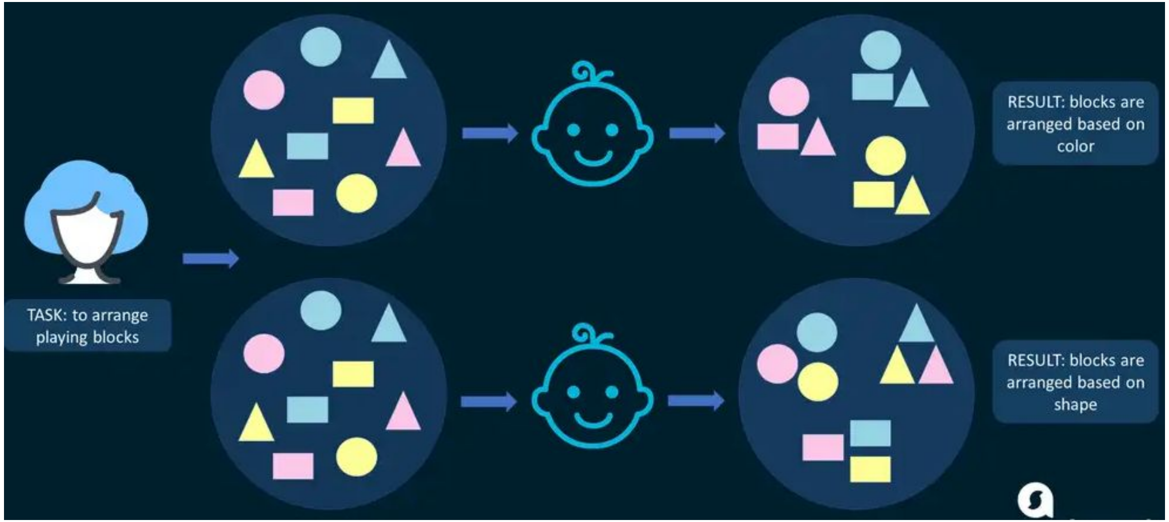
Advantage:

- Result can be useful as a pre-processing step for supervised learning

Unsupervised Learning

Classification and Regression are important tasks belonging to the "supervised" realm. But there are many other tasks where the unsupervised learning can help:

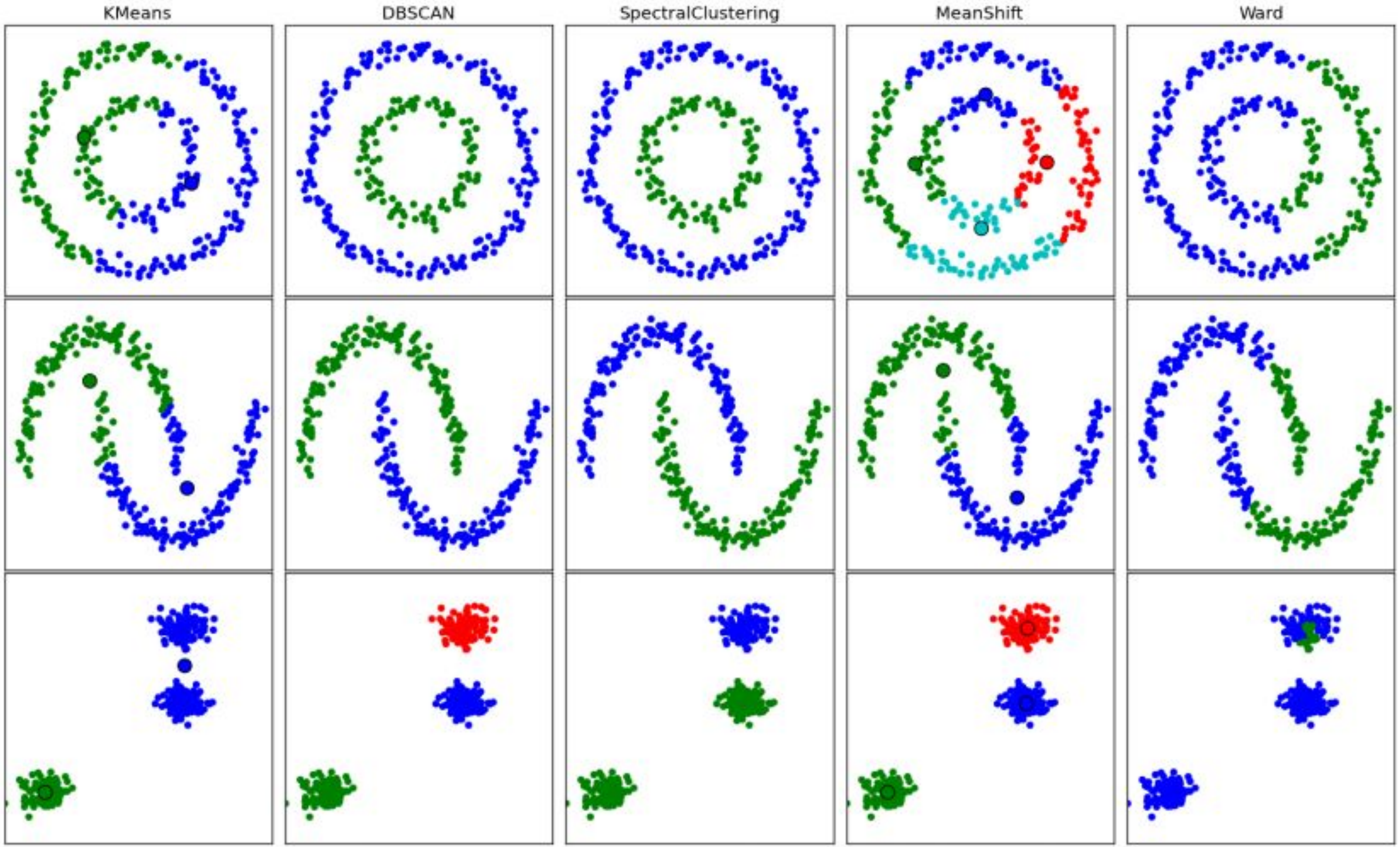
- Clustering
- Association rules
- Dimensionality reduction


TASK: to arrange playing blocks
RESULT: blocks are arranged based on color
RESULT: blocks are arranged based on shape

Clustering

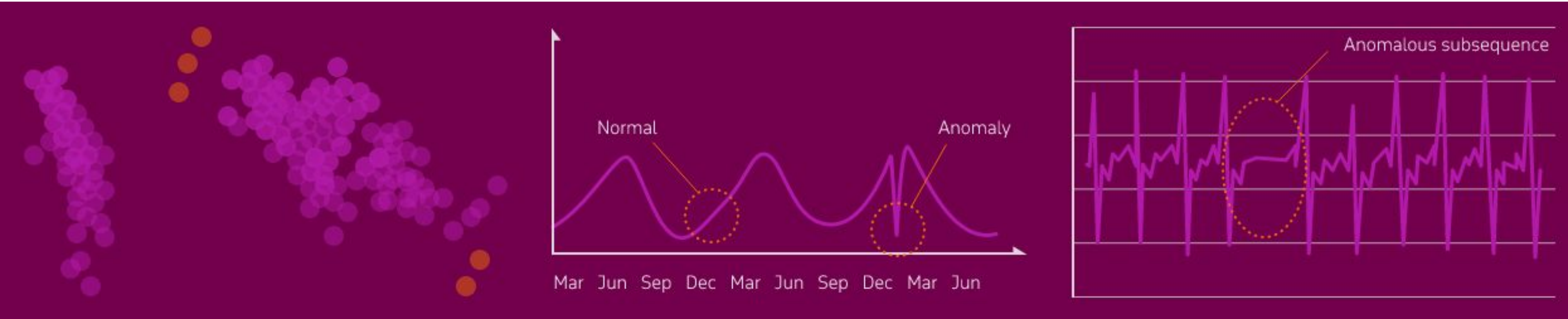
- Find structures in the data, organize by similarity
- An ML model finds any patterns, similarities, and/or differences within uncategorized data structure by itself
- Often can be a useful input to other tasks
- Real-life applications:
    - **Customer and market segmentation:** can help group people that have similar traits and create customer personas for more efficient marketing and targeting campaigns
    - **Clinical cancer studies:** used to study cancer gene expression data (tissues) and predict cancer at early stages
    - **Anomaly detection**

Clustering



Anomaly detection

- With clustering, it is possible to detect any sort of outliers in data
- For example, companies engaged in transportation and logistics may use anomaly detection to identify logistical obstacles or expose defective mechanical parts
- Financial organizations may utilize the technique to spot fraudulent transactions and react promptly, which ultimately can save lots of money
- From the modeling of purchasing habits; or new physics searches!



Clustering types

Exclusive clustering

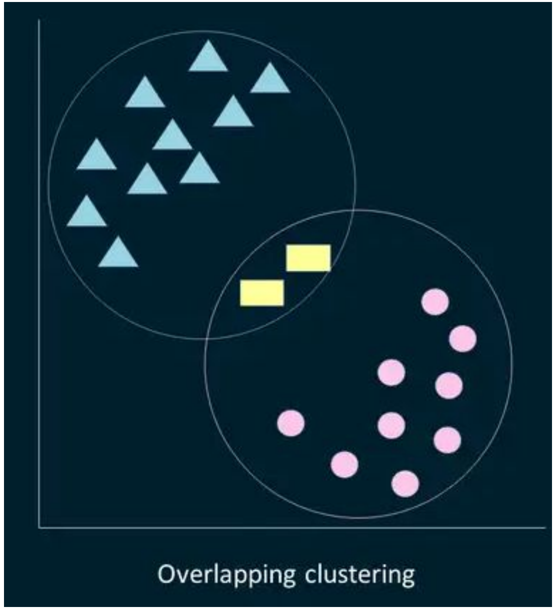
- A “hard” clustering where a grouping in which one piece of data can belong only to one cluster
- **Using mostly K-means:** an algorithm for exclusive clustering, also known as partitioning or segmentation
- It puts the data points into the predefined number of clusters known as K
- K in the K-means algorithm is the input since you tell the algorithm the number of clusters you want to identify in your data
- Each data item then gets assigned to the nearest cluster center, called centroids and the latter act as data accumulation areas

K-Means: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Clustering types

Overlapping clustering:

- “Soft” clustering allows data items to be members of more than one cluster with different degrees of belonging
- Fuzzy K-means are often used in this case: an extension of the K-means algorithm used to perform overlapping clustering
- Unlike the K-means algorithm, it implies that data points can belong to more than one cluster with a certain level of closeness towards each
- The closeness is measured by the distance from a data point to the centroid of the cluster



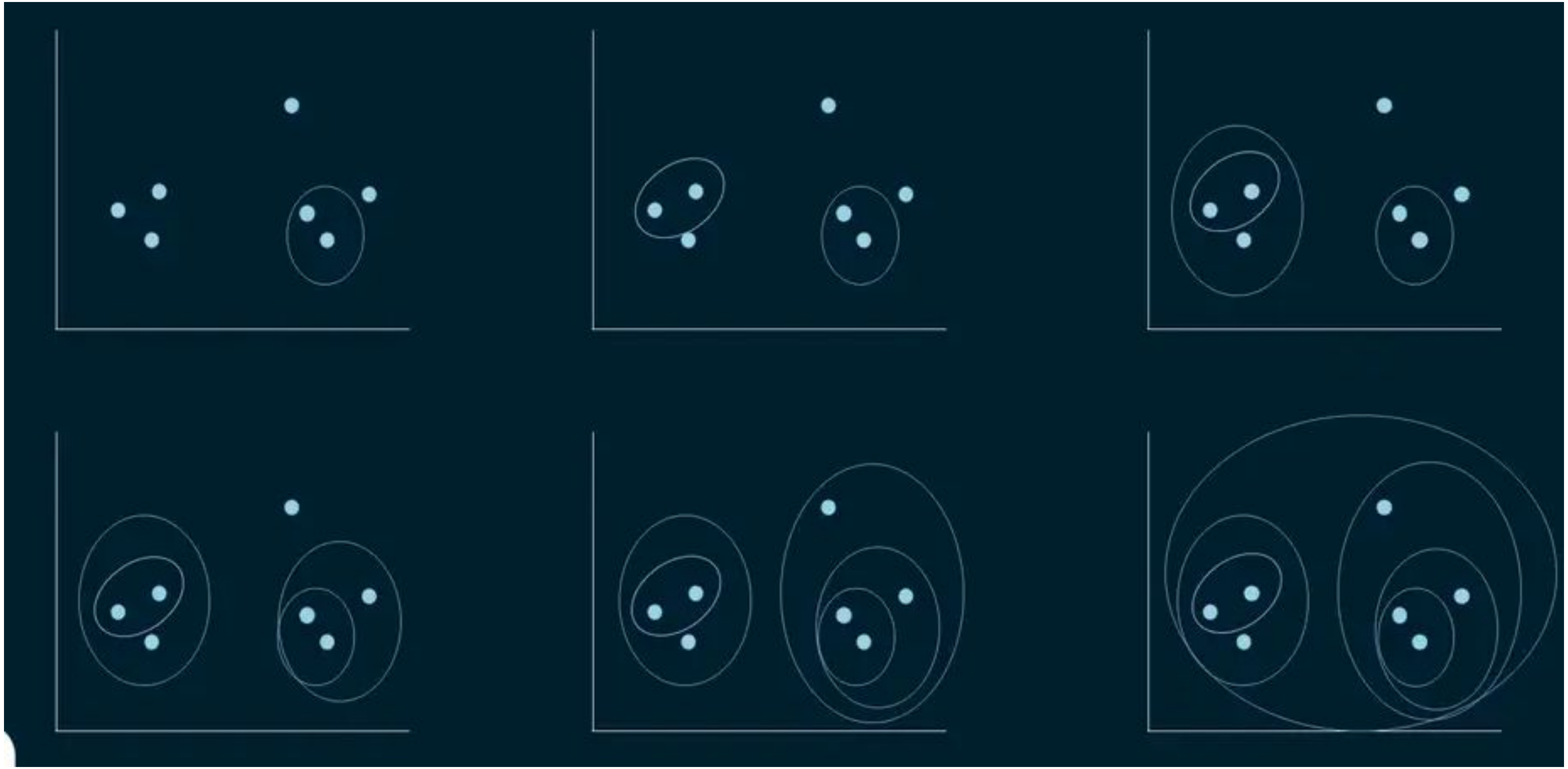
Clustering types

Hierarchical clustering:

- Creating a hierarchy of clustered data items
- To obtain clusters, data items are either decomposed or merged based on the hierarchy
- Such approach may start with each data point assigned to a separate cluster
- Two clusters that are closest to one another are then merged into a single cluster
- The merging goes on iteratively till there's only one cluster left at the top
- Such an approach is known as bottom-up or agglomerative

Clustering types

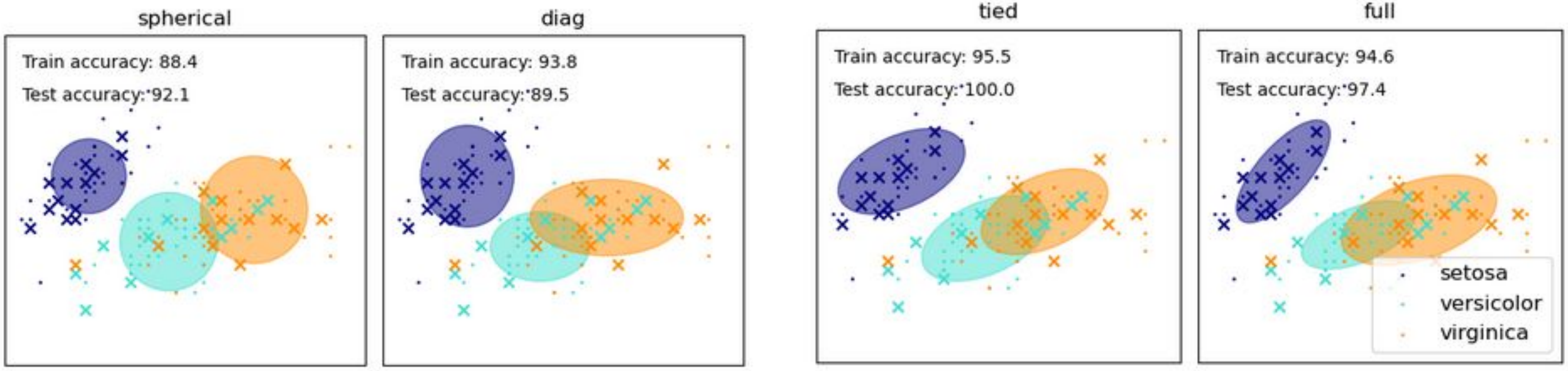
Hierarchical clustering:



Clustering types

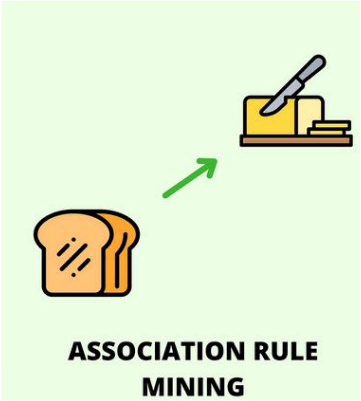
Gaussian Mixture Models

- Used in probabilistic clustering (diagonal, spherical, tied and full covariance matrices supported)
- Since the mean or variance is unknown, the models assume that there is a certain number of Gaussian distributions, each representing a separate cluster
- The algorithm is basically used to decide which cluster a particular data point belongs to



Association rules

- Rule-based unsupervised learning method aimed at discovering relationships and associations between different variables in large-scale datasets
- The rules present how often a certain data item occurs in datasets and how strong and weak the connections between different objects are
- Widely used to analyze customer purchasing habits, allowing companies to understand relationships between different products and build more effective business strategies



Dimensionality reduction

- Sometimes, the number of dimensions gets too high, resulting in the performance reduction of ML algorithms and data visualization hindering

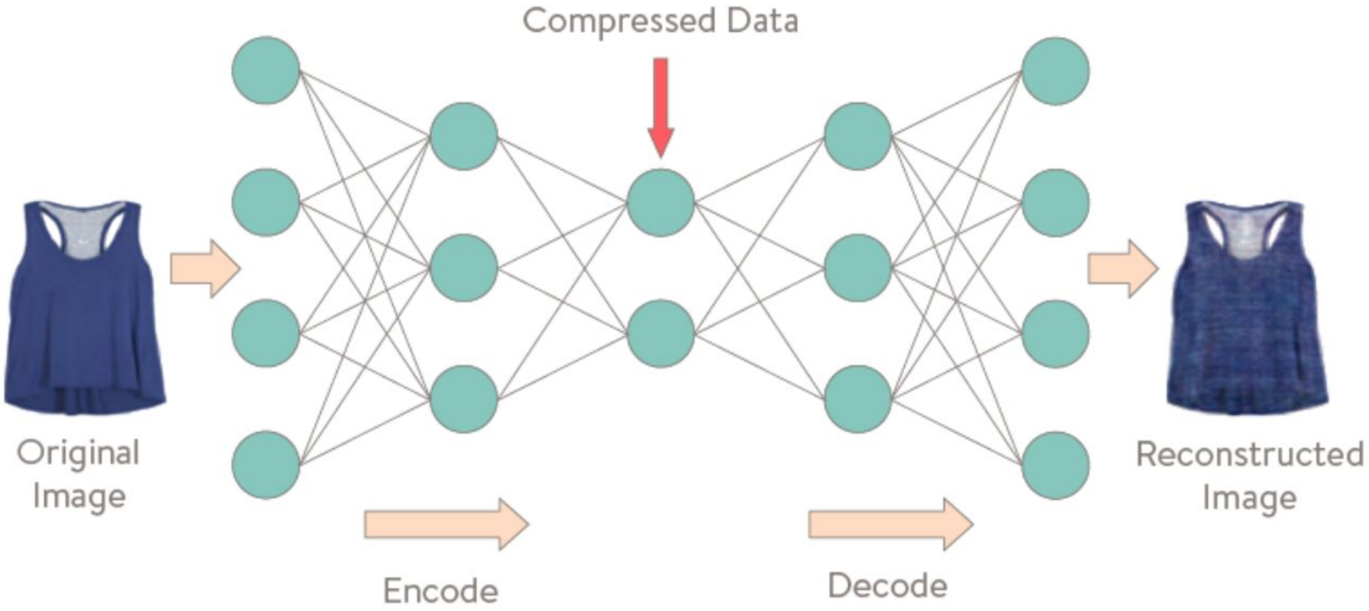
- It makes sense to reduce the number of features – or dimensions – and include only relevant data ⇒ **Dimensionality reduction**

- Number of data inputs becomes manageable while the integrity of the dataset is not lost

Density-based approach:
https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Auto encoders

- Compressed data is treated as a result of an algorithm
- Trained representations can be made stable against different noise



Auto encoders architecture

1- **Encoder:** In which the model learns how to reduce the input dimensions and compress the input data into an encoded representation

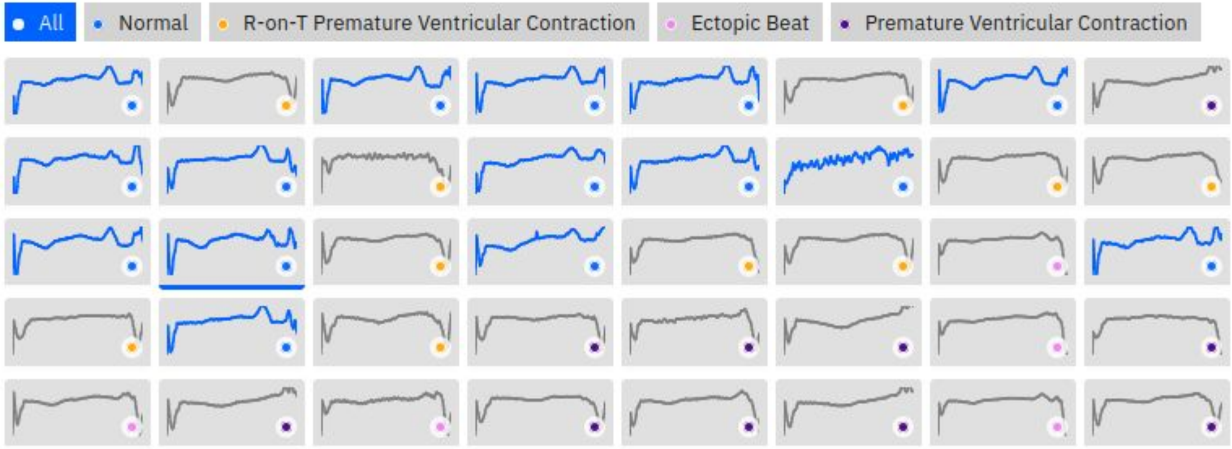
2- **Bottleneck:** which is the layer that contains the compressed representation of the input data (the lowest possible dimensions of the input data)

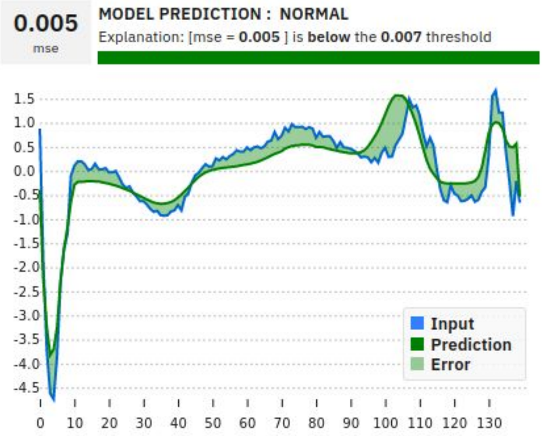
3- **Decoder:** In which the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible

4- **Reconstruction Loss:** This is the method that measures measure how well the decoder is performing and how close the output is to the original input

Auto encoders applications

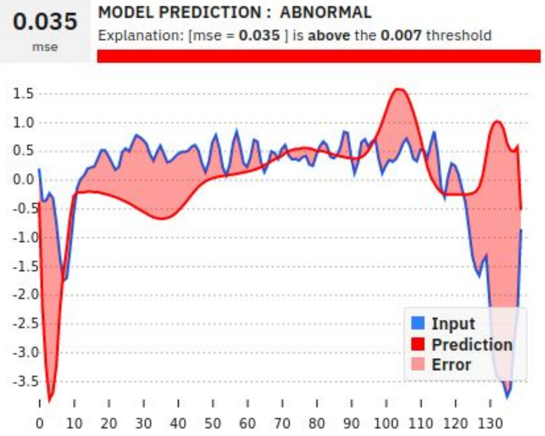
- Detect anomalies in electrocardiograms ⇒ Play it here
- Anomalies in ISS Telemetry Data with Airbus


All
Normal
R-on-T Premature Ventricular Contraction
Ectopic Beat
Premature Ventricular Contraction


0.005 mse

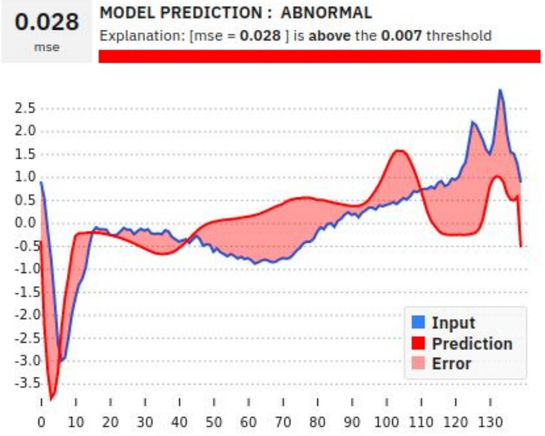
MODEL PREDICTION : NORMAL

Explanation: [mse = **0.005**] is **below** the **0.007** threshold


0.035 mse

MODEL PREDICTION : ABNORMAL

Explanation: [mse = **0.035**] is **above** the **0.007** threshold

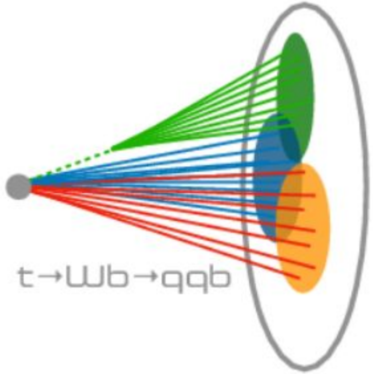

0.028 mse

MODEL PREDICTION : ABNORMAL

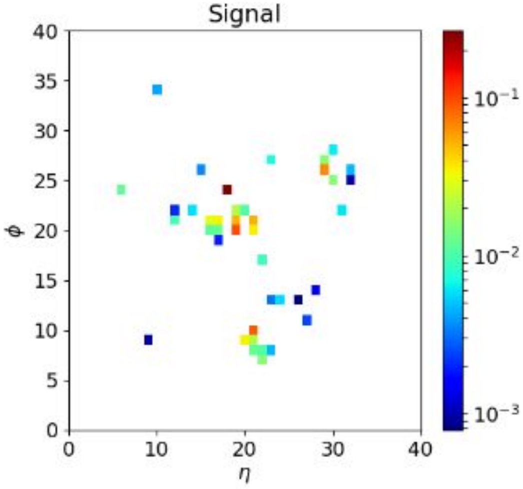
Explanation: [mse = **0.028**] is **above** the **0.007** threshold

Auto encoders applications

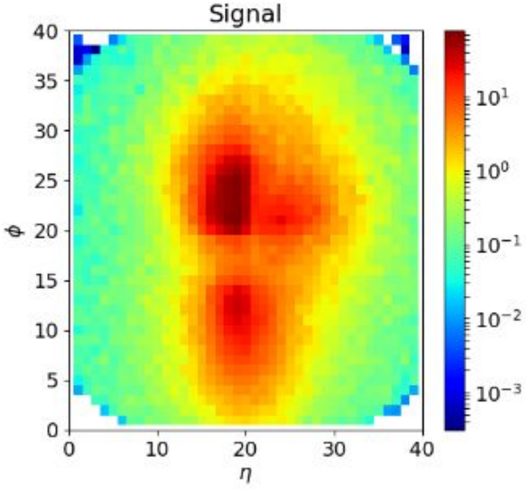
- On HEP, one example would be the top-tagging



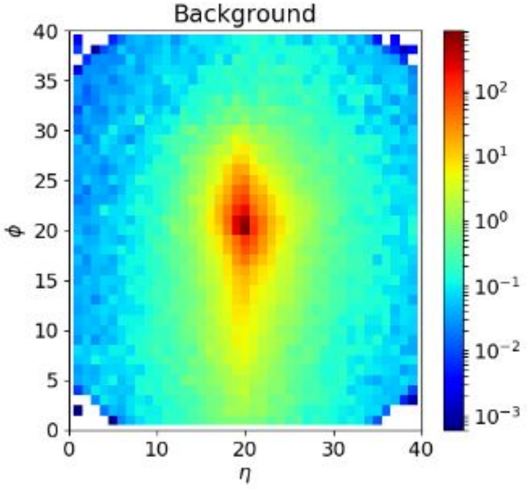
Typical single top-jet



Average top-jet

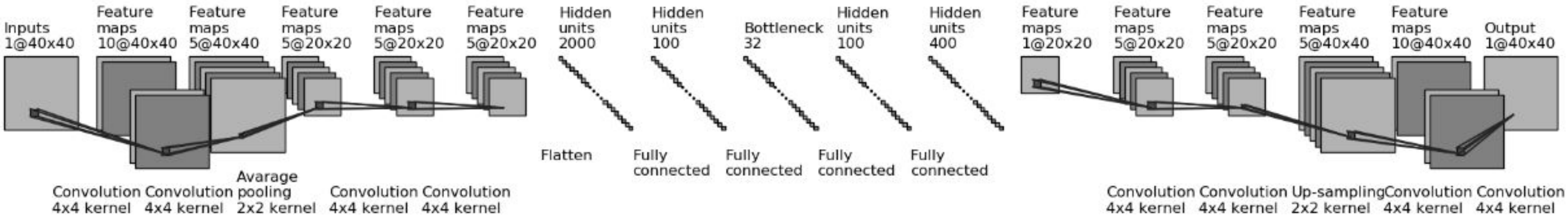


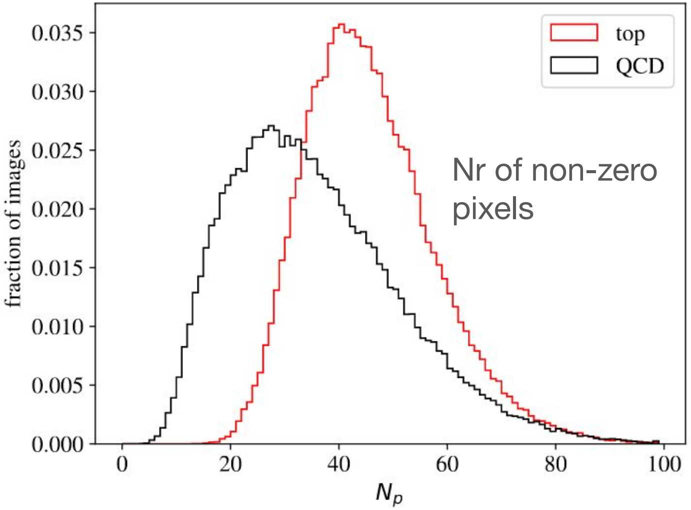
Average QCD-jet



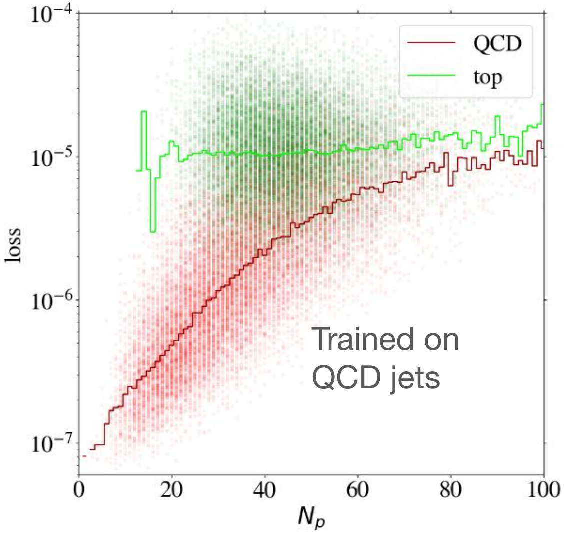
Auto encoders applications

- On HEP, one example would be the top-tagging

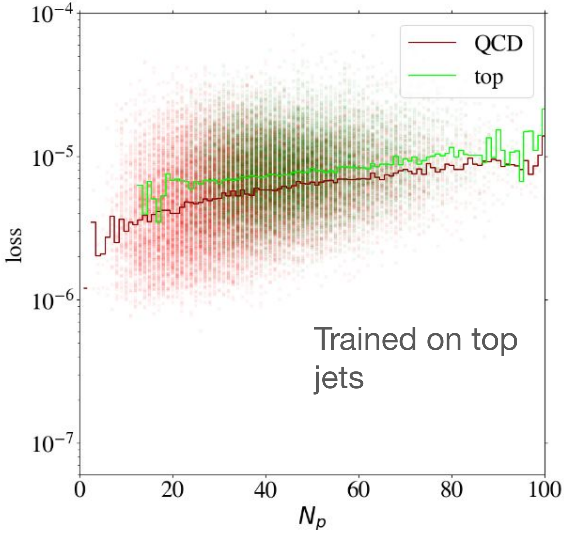




Nr of non-zero pixels



Trained on QCD jets



Trained on top jets

Limitations

To supervised learning:

- Slow (it requires human experts to manually label training examples one by one)
- Costly (a model should be trained on the large volumes of hand-labeled data to provide accurate predictions)

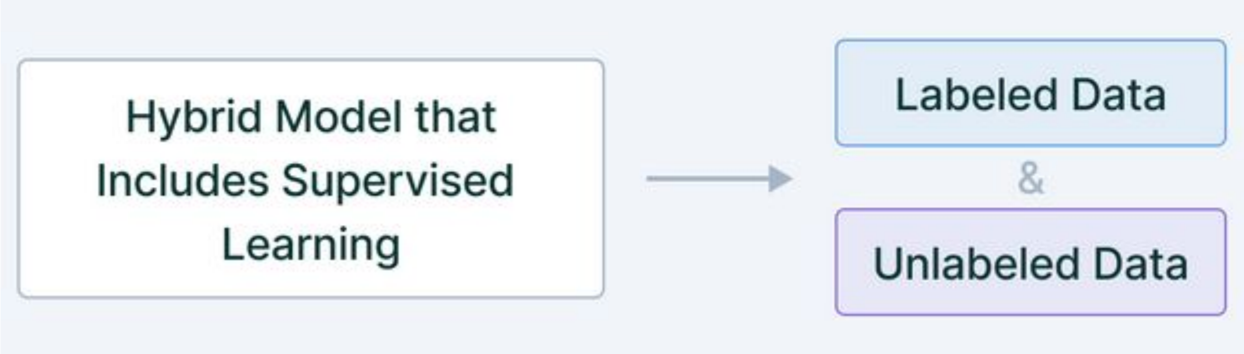
To unsupervised learning:

- Has a limited area of applications (mostly for clustering purposes)
- Provides less accurate results

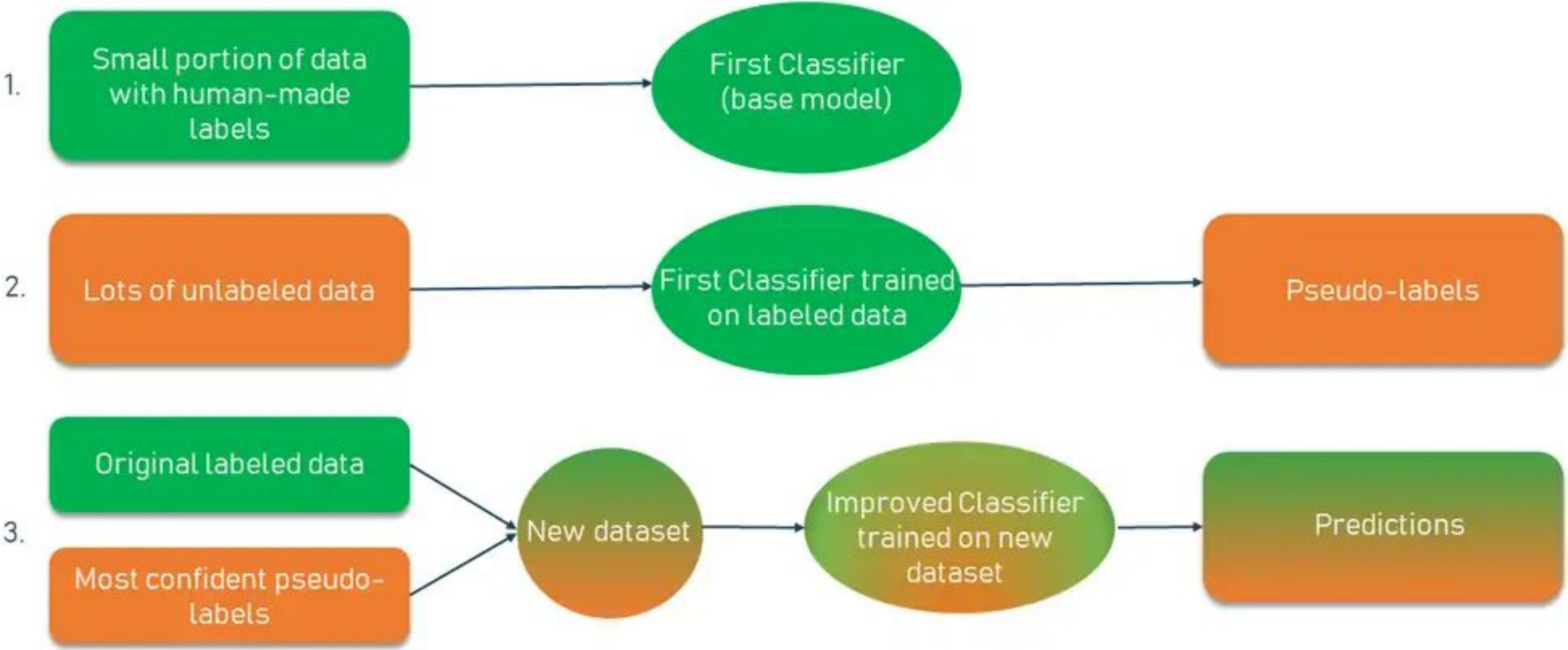
Semi-supervised Learning

- It is used in scenarios where we have access to large amounts of data, and only a small portion of that is labeled. The more (relevant) data we use for training, the more robust our model becomes.

- Semi-Supervised Learning works by initially training the model using the labeled dataset, just like Supervised Learning. Once we get the model performing well, we use it to predict the remaining unlabeled data points and label them with the corresponding predictions.



Self-training



1. Small portion of data with human-made labels

First Classifier (base model)

2. Lots of unlabeled data

First Classifier trained on labeled data

Pseudo-labels

3. Original labeled data

Most confident pseudo-labels

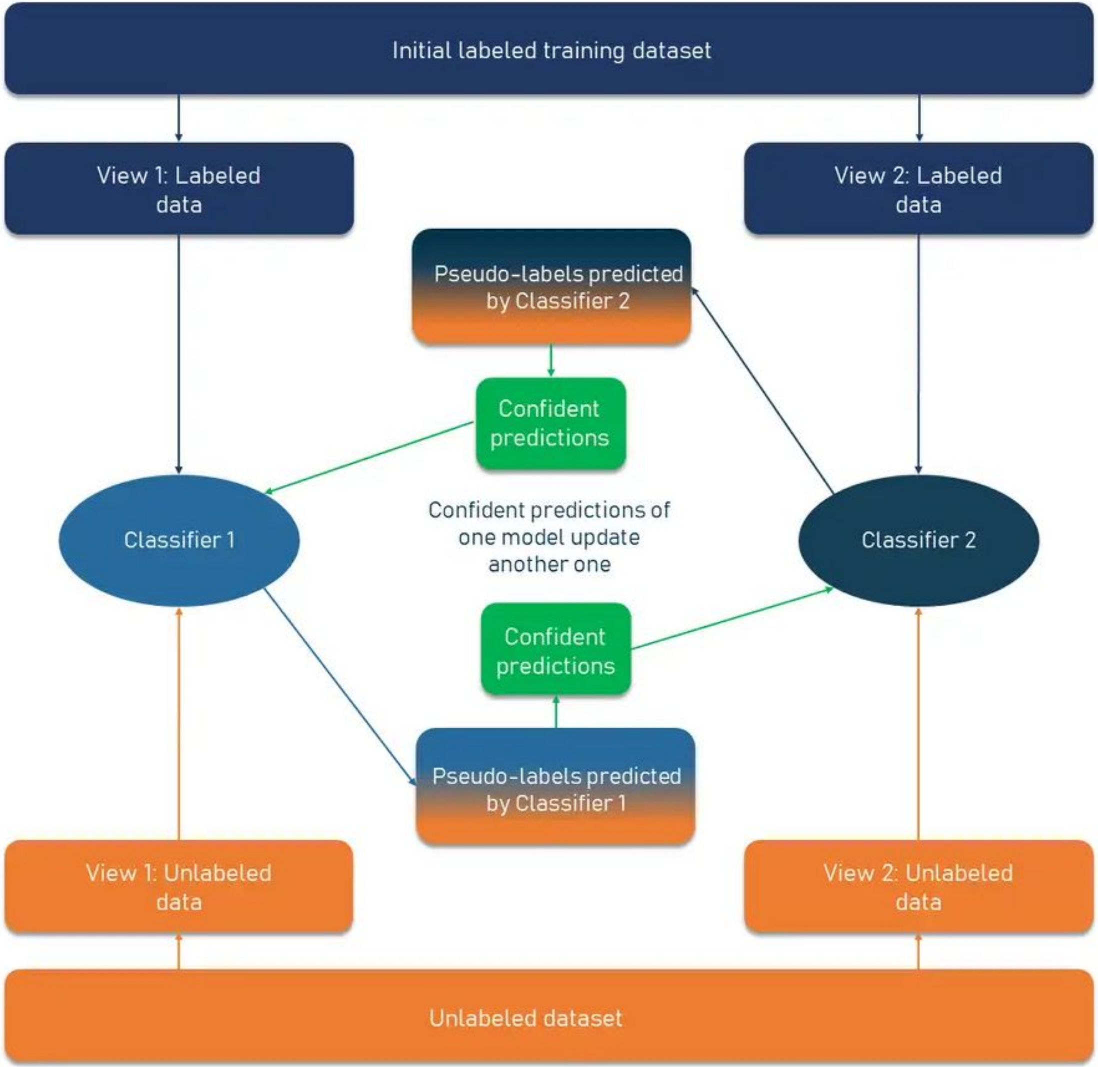
New dataset

Improved Classifier trained on new dataset

Predictions

Co-training

- Used when only a small portion of labeled data is available
- Trains two individual classifiers based on two views of data
- The class of sample data can be accurately predicted from each set of features alone



Co-training

In simple terms:

- First, you train a separate classifier (model) for each view with the help of a small amount of labeled data

- Then the bigger pool of unlabeled data is added to receive pseudo-labels

- Classifiers co-train one another using pseudo-labels with the highest confidence level. If the first classifier confidently predicts the genuine label for a data sample while the other one makes a prediction error, then the data with the confident pseudo-labels assigned by the first classifier updates the second classifier and vice-versa.

- The final step involves the combining of the predictions from the two updated classifiers to get one classification result.

Transformers

- Special case of a graph neural network
- Self attention is implemented via message passing on the fully connected graph
- Strength comes from:
  - Multi-headed attention operation
  - Dense network node updates
  - More efficient to train than a full GNN

- Widely used in natural language processing
  - Introduced in 2017 by a team at Google Brain to replace RNNs
  - e.g. DALL-E 2 is also transformer based

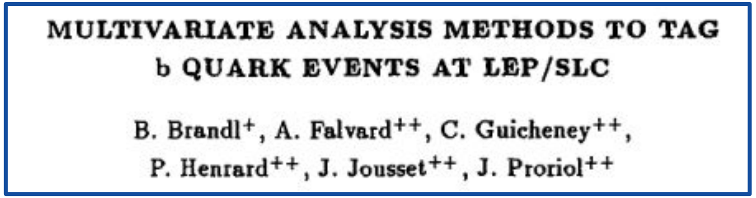
Transformers

- Special case of a graph neural network
- Self attention is implemented via message passing on the fully connected graph
- Strength comes from:
  - Multi-headed attention operation
  - Dense network node updates
  - More efficient to train than a full GNN

- Widely used in natural language processing
  - Introduced in 2017 by a team at Google Brain to replace RNNs
  - e.g. DALL-E 2 is also transformer based

Our take with DALLE-2: *oil painting of Transformers at CERN*

Transformers

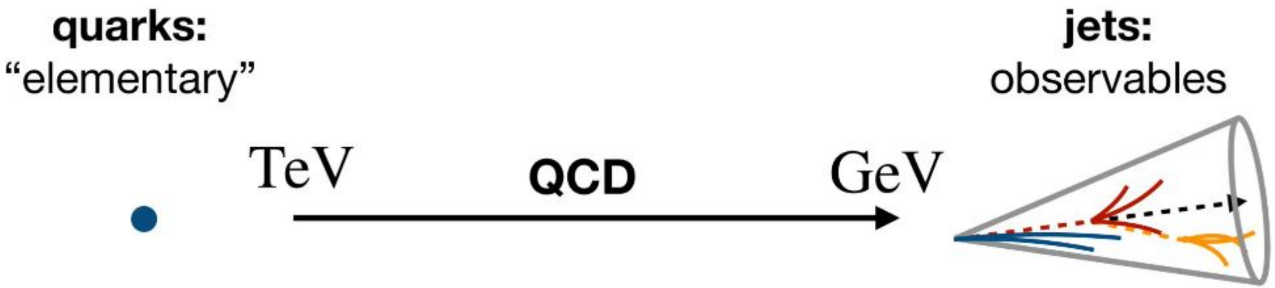


MULTIVARIATE ANALYSIS METHODS TO TAG b QUARK EVENTS AT LEP/SLC

B. Brandl^{+}, A. Falvard^{++}, C. Guicheney^{++},
P. Henrard^{++}, J. Jousset^{++}, J. Proriol^{++}

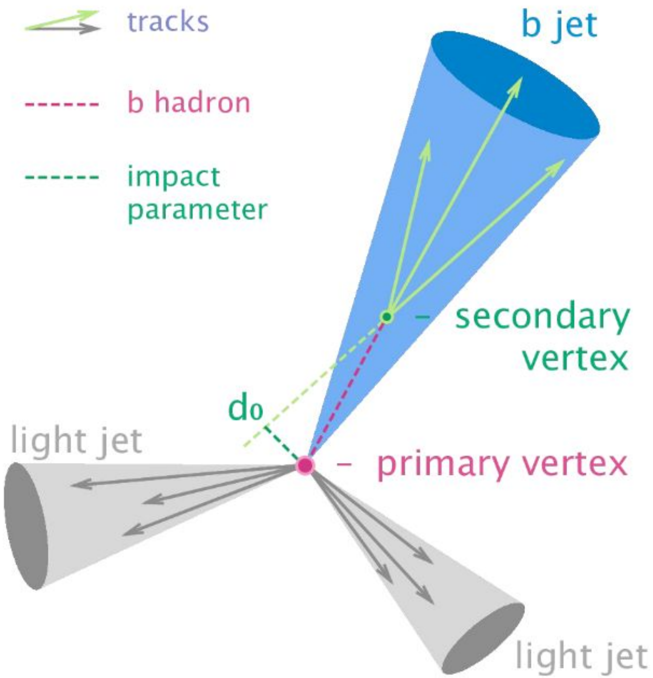
An example from flavour tagging in ATLAS:

From 1992!



- After the hard interaction, quarks hadronise into hadrons
- Heavy hadrons can decay into lighter hadrons/leptons
→ Cascade-like behaviour
- Goal of (heavy) flavour tagging → Identify the type of the quark which instantiated the jet
- Focusing on the identification of jets initiated by b-jets (aka b-tagging)

Transformers



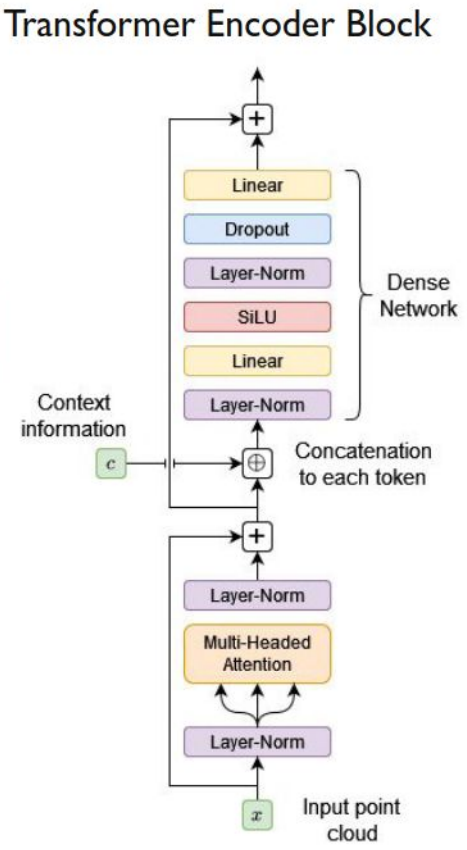
Use the topology of heavy-flavour jets

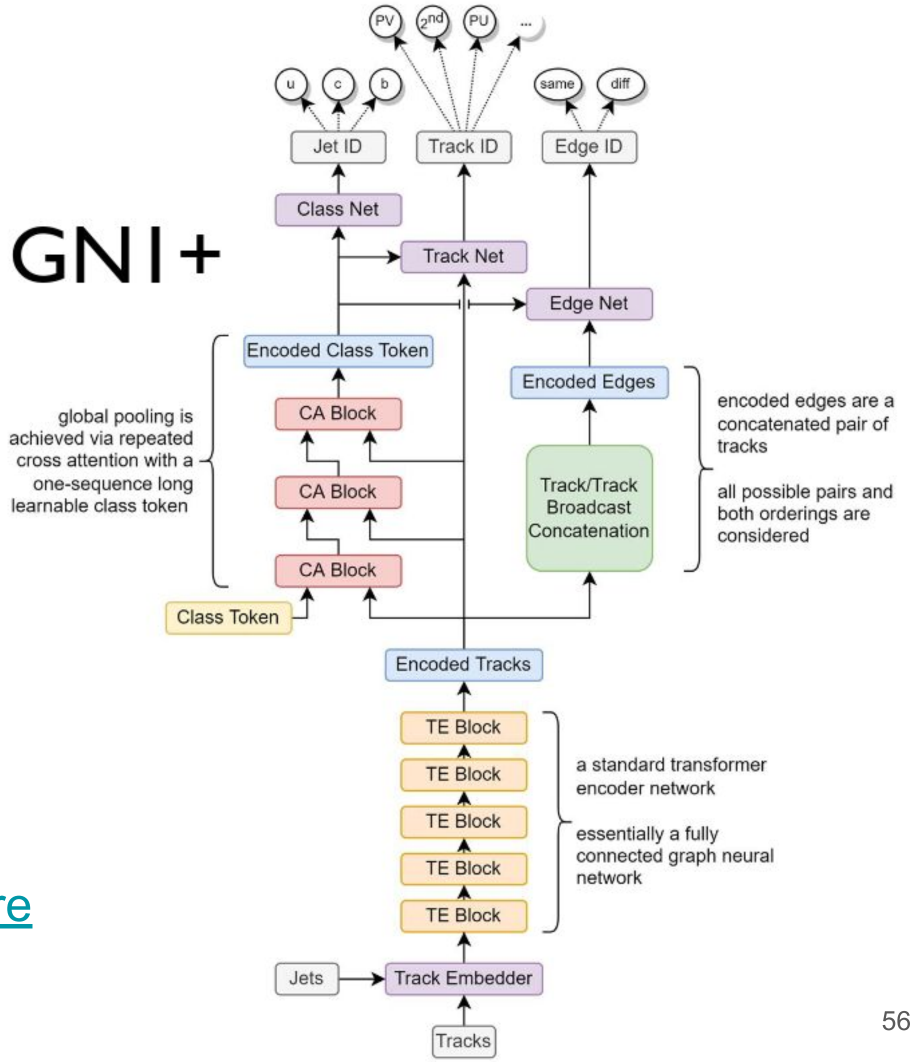
- Lifetime of b-hadrons (c · τ ≈ 5mm at pT = 50 GeV)

Information about the topology is provided via track and jet variables

- Track variables
  - Impact parameter (d0/z0)
  - Number of inner detector hits
  - Fraction of pT coming from the track
- Jet variables
  - Jet kinematics (pT, |η|)

Transformers


Transformer Encoder Block



global pooling is achieved via repeated cross attention with a one-sequence long learnable class token

encoded edges are a concatenated pair of tracks

all possible pairs and both orderings are considered

a standard transformer encoder network

essentially a fully connected graph neural network

More information here

Semi-supervised learning examples

- **Speech recognition:** Labeling audio is a very resource- and time-intensive task, so semi-supervised learning can be used to overcome the challenges and provide better performance (as Meta)

- **Web content classification:** With billions of websites presenting all sorts of content out there, classification would take a huge team of human resources to organize information on web pages by adding corresponding labels (example of Google search)

- **Text document classification:** Training a train a base long short-term memory model on a few text examples with hand-labeled most relevant words and then apply it to a bigger number of unlabeled samples

Conclusions

- **Supervised:** All data is labeled and the algorithms learn to predict the output from the input data ⇒ You know what you are looking for in data and provides more accurate results

- **Unsupervised:** All data is unlabeled and the algorithms learn to inherent structure from the input data ⇒ Results may be less accurate and training process is relatively time-consuming because algorithms need to analyze and calculate all existing possibilities

- **Semi-supervised:** Some data is labeled but most of it is unlabeled and a mixture of supervised and unsupervised techniques can be used ⇒ Promising results in classification tasks with a minimal amount of labeled data and plenty of unlabeled data

Thanks for the attention!